

ÉCOLE CENTRALE DE LYON

UE PRO

PROJET D'APPLICATION RECHERCHE N° 140 :  
ANALYSE ET PRÉDICTION DE DONNÉES SPATIO-TEMPORELLES SPORTIVES

---

## Rapport RVP 2

---

*Auteurs :*

Arthur MAIA MENDES  
Emeline GOT

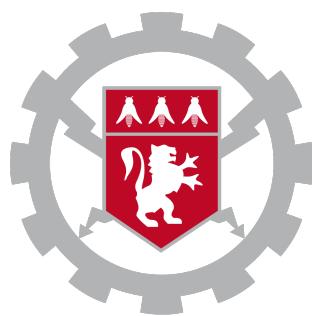
*Tuteur :*

Romain VUILLEMOT

*Conseiller :*

Vincent FRIDRICI

28 janvier 2017



# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Objectifs</b>	<b>2</b>
<b>3</b>	<b>Livrable</b>	<b>2</b>
<b>4</b>	<b>Organisation</b>	<b>2</b>
<b>5</b>	<b>Outils</b>	<b>4</b>
5.1	Présentation de la plateforme AMIGO . . . . .	4
5.2	Le logiciel VICON . . . . .	5
5.3	Moyens de visualisation . . . . .	5
5.4	Python . . . . .	6
5.5	Blender . . . . .	7
<b>6</b>	<b>Études des modèles sur les trajectoires 3D</b>	<b>8</b>
6.1	Reconnaissance de formes . . . . .	8
6.2	Problème rencontré . . . . .	9
<b>7</b>	<b>Simplification du problème : trajectoires 2D</b>	<b>9</b>
7.1	Principe général . . . . .	9
7.2	Base de données MNIST . . . . .	10
7.3	Régression Softmax - Modèle mathématique . . . . .	10
7.4	Implémentation Python du modèle . . . . .	12
7.5	Prédiction . . . . .	13
<b>8</b>	<b>Conclusion</b>	<b>16</b>
<b>A</b>	<b>Diagramme de Gantt</b>	<b>18</b>
<b>B</b>	<b>Rapport d'activité</b>	<b>20</b>

# 1 Introduction

Le projet d'application recherche (PAr) s'encadre dans l'unité d'enseignement professionnelle de l'École Centrale de Lyon. Pour cette action de formation deux étudiants, sous la supervision d'un tuteur et d'un conseiller enseignants de l'école, sont censés développer un projet de recherche dans un des domaines traités dans l'école.

Le projet traité dans ce rapport est intitulé "Analyse et prédition de données spatio-temporelles sportives". Les sports génèrent de très grands volumes de données, qui sont pour le moment sous-exploitées. Par exemple au football, même si on connaît en temps réel la position des joueurs et du ballon, seules des analyses simples sont calculées comme la distance parcourue des joueurs, ou la possession par équipe. Ces données sont pourtant très riches et pourraient permettre de générer automatiquement des tactiques d'équipes, et prédire les prochaines actions ou comportements des joueurs.

Ces données spatio-temporelles ne sont pas évidentes à explorer visuellement, et encore moins à communiquer car souvent concentrées sur un espace limité (un terrain), et mises à jour en continue (au fil de chaque action). Même s'il serait possible d'abstraire et de simplifier ces données (sous forme de séries temporelles par exemple), l'information spatiale en reste difficilement dissociable ; ainsi les dimensions espace et temps doivent être conservées et analysées conjointement [1].

# 2 Objectifs

Le but de ce PAr est de trouver et d'expérimenter des techniques de modélisation et de prévisions de données issues de Motion Capture (3D) et/ou spatio-temporelles (2D). Il sera ensuite question de créer une interface de visualisation pour représenter les résultats de ces données et prédictions (prototypes papiers, vidéos concepts, prototype interactif). Cette interface de visualisation pourra être accompagnée d'un guide d'utilisation si sa création est aboutie afin de permettre à d'autres personnes de travailler facilement dessus.

# 3 Livrable

Le livrable consiste dans un modèle de prévision des mouvements humains. On souhaite obtenir celui-ci et y accéder via une interface de visualisation que l'on doit concevoir. Celle-ci peut être sous la forme d'un site internet, d'une vidéo, ou bien d'un logiciel. Nous travaillons sur la visualisation avec les données que nous avons enregistrées pour le bras droit. En parallèle, nous souhaitons trouver un modèle de prédition que nous pouvons appliquer sur des trajectoires simples dans un premier temps. Une fois le modèle validé, nous souhaitons développer une interface d'utilisation simple où les différents mouvements possibles à partir d'un morceau de trajectoire seraient accessibles en temps réel.

# 4 Organisation

Pour ce qui est de l'organisation, nous avons déterminé trois tâches principales pour ce projet. La première a déjà été effectuée et concerne l'exploitation des données enregistrées à l'aide de la plateforme AMIGO. La seconde s'intéresse plutôt à l'ensemble de la partie mathématique

de ce projet qui consiste en trouver un modèle de prévision applicable pour la prédiction de mouvement. Et enfin nous devrons réaliser l'interface de visualisation de la plateforme en accord avec les attentes du tuteur et des éventuelles utilisations futures par d'autres projets. S'ajoute à cela une tâche de gestion de projet avec la rédaction des différents livrables et la préparation des soutenances de RVP et de fin de projet. La Figure 1 présente une version simplifiée du diagramme PERT du projet.

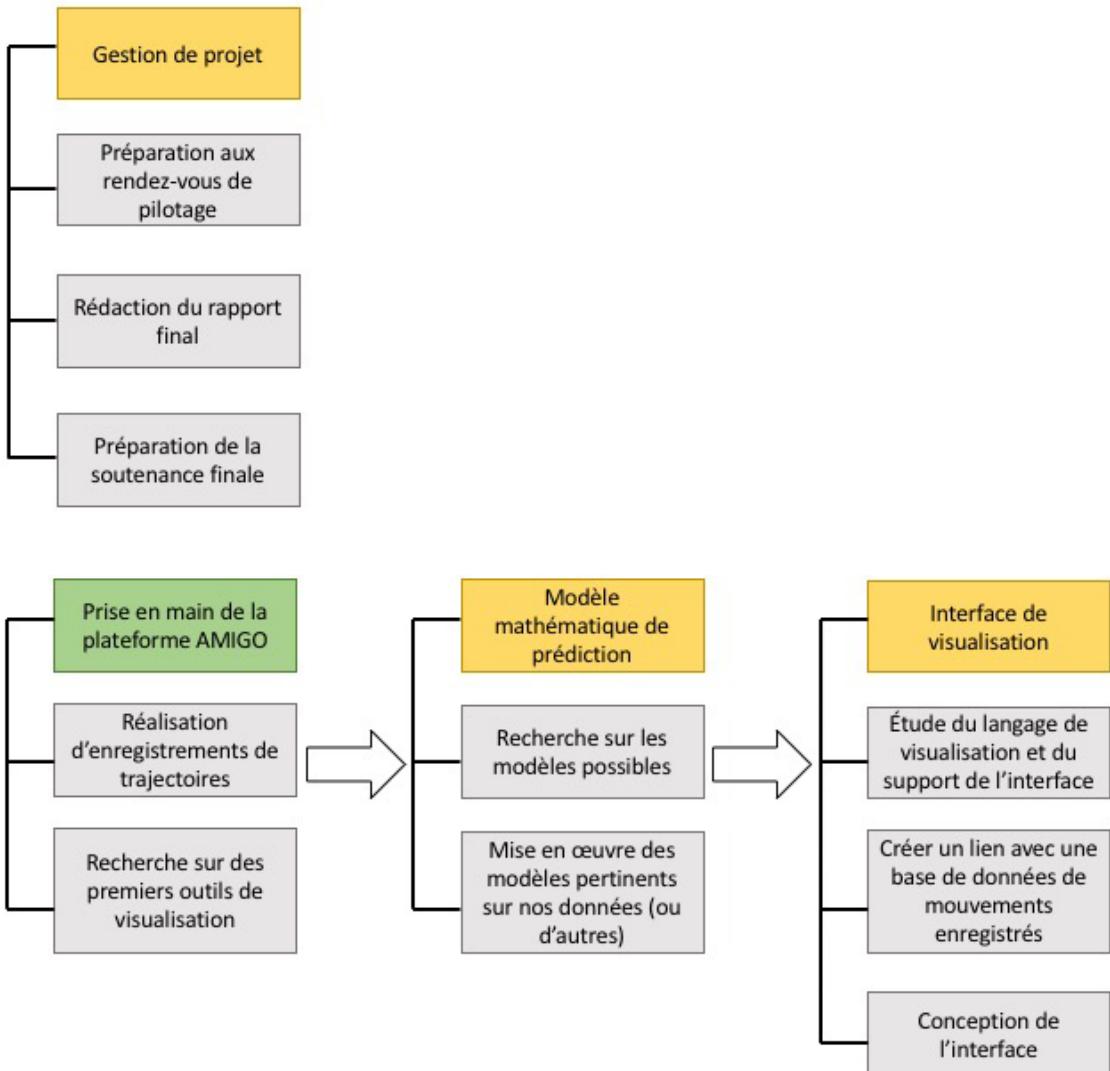


FIGURE 1 – Tâches principales et sous-tâches associées

Le diagramme de Gantt associé est présent en Annexe A.

Nous travaillons également avec un rapport d'activité (Annexe B) qui nous permet de rester en contact avec notre tuteur et qu'il soit capable de suivre notre avancée. Celui-ci tient à jour

les activités effectuées chaque semaine, la bibliographie trouvée, ainsi que les remarques et les problèmes rencontrés.

## 5 Outils

Nous avons dans un premier temps choisi de travailler avec un mouvement 3D afin d'utiliser des données obtenues avec la plateforme AMIGO de Motion Capture.

### 5.1 Présentation de la plateforme AMIGO

L'École Centrale de Lyon a récemment fait l'acquisition d'une plateforme de Motion Capture mise à disposition en collaboration avec la région Auvergne - Rhône-Alpes.

Cette plateforme permet l'enregistrement de mouvements 3D à l'aide d'un dispositif de Motion Capture avec des caméras de la marque VICON [2]. Celui-ci est installé dans une pièce d'environ  $20m^2$ .

L'enregistrement s'effectue à l'aide de 6 caméras disposées de part et d'autre de la pièce. Ce sont des caméras à infrarouges qui ont pour but d'enregistrer uniquement les mouvements de capteurs appelés réflecteurs.

Ces réflecteurs sont des petites sphères grises qui peuvent être placées facilement à différents endroits du corps à l'aide de double face (Figure 2). Il est intéressant de placer les capteurs d'une certaine manière sur les membres du corps afin de pouvoir enregistrer entièrement un mouvement à l'aide des caméras. Par exemple, il faudra faire attention à ce que les capteurs ne soient pas masqués par les vêtements lors du mouvement.



FIGURE 2 – Exemple de placement de réflecteurs sur un bras

## 5.2 Le logiciel VICON

Pour enregistrer les données de mouvement, les caméras sont reliées à l'ordinateur qui utilise le logiciel VICON associé : Nexus [3]. Après une phase d'initialisation du logiciel où nous devons effacer les parasites captés par les caméras et mettre à jour l'origine de l'enregistrement, nous pouvons effectuer l'enregistrement du mouvement qui se traduit par le mouvement des différents capteurs présents dans l'espace enregistré par les caméras.

Le logiciel (Figure 3) nous permet de traiter ces résultats afin de compléter les bouts de trajectoire manquants par exemple. Cela passe par la définition de contraintes sur les capteurs liées à l'anatomie du sujet observé. Nous pouvons définir des segments rigides entre les capteurs et ensuite préciser les liaisons qui existent entre ces différents segments. Cela permet en somme de définir un véritable squelette qui permet de mieux comprendre le mouvement des capteurs les uns par rapport aux autres. Nous pouvons également enregistrer ce squelette et le réutiliser plusieurs fois pour faire des enregistrements sur le même sujet avec les capteurs positionnés au même endroit.

La première phase de détermination du squelette peut s'avérer assez longue et fastidieuse. Généralement, il faut compter entre 30 et 45 minutes pour calibrer correctement un squelette complet. Mais une fois effectuée, l'enregistrement des données est très simplifié et il n'y a plus besoin de repasser par cette étape pour les enregistrements suivants.

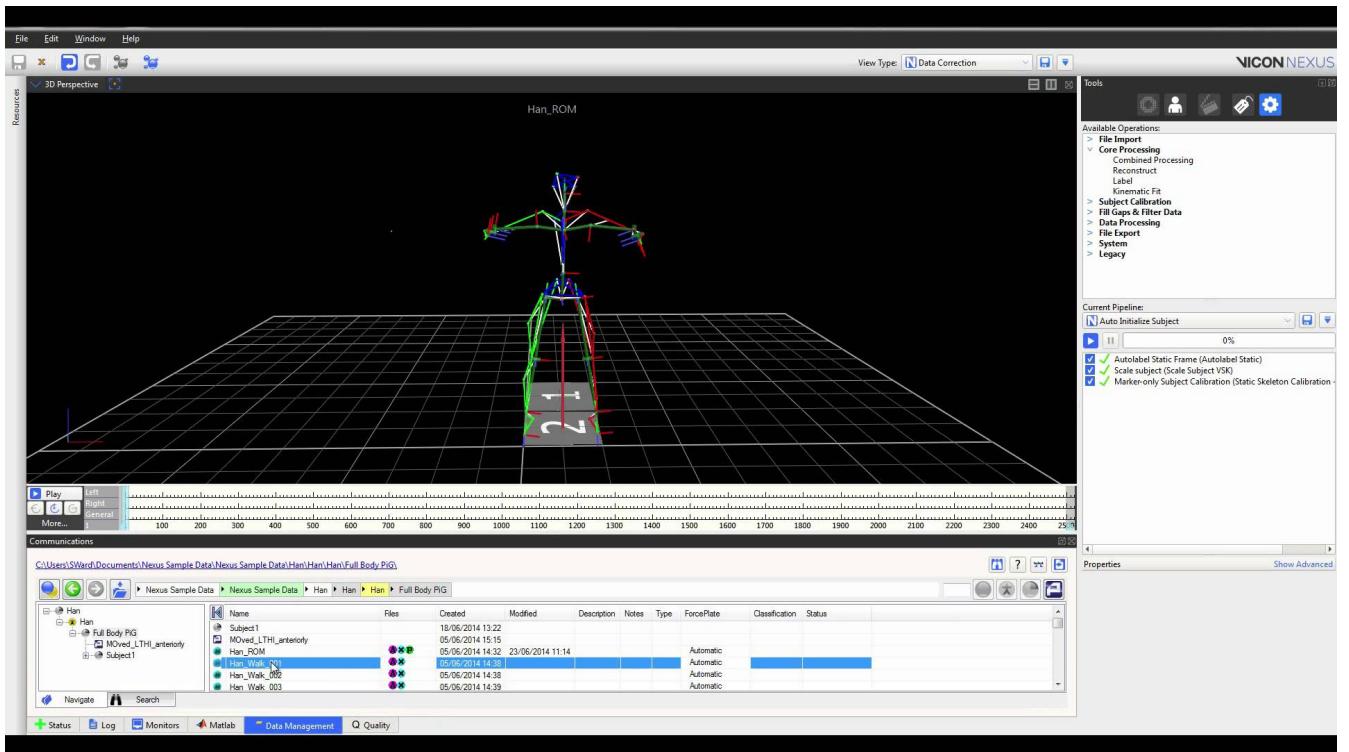


FIGURE 3 – Interface du logiciel NEXUS

## 5.3 Moyens de visualisation

Une fois l'enregistrement effectué à l'aide de la plateforme et du logiciel d'acquisition, nous avons cherché un moyen de visualiser les données.

Divers logiciels ou codes permettent de visualiser des données 3D, parmi les plus utilisés :

- Le plug-in Maya de Autodesk
- Python avec la bibliothèque Matplotlib
- Catia
- Plotly (représentation de courbes en ligne)
- Notebook Python
- Format JSON avec JavaScript
- Blender

Nous avons travaillé avec ce plugin de python comme c'est également ce qui est utilisé par le second PAr de Motion Capture pour visualiser les données. Nous partageons avec eux le code que nous avons mis en place et nous avons accès à leurs modifications hebdomadaires concernant la mise en place d'une visualisation plus poussée que celle que nous recherchons via une plateforme Github [4]. La plateforme Github permet de mettre en commun des codes informatiques et de permettre une collaboration pour apporter des modifications ou des améliorations pertinentes aux codes d'autres personnes. En collaboration avec le PE n° 41 (MoCap et synthèse 3D), nous avons commencé à utiliser le logiciel gratuit Blender [5] pour réaliser la visualisation des trajectoires enregistrées sur la plateforme AMIGO.

## 5.4 Python

Avec le code Python et le module Matplotlib, nous sommes capables d'obtenir une visualisation d'un capteur dans l'espace 3D (Figure 4).

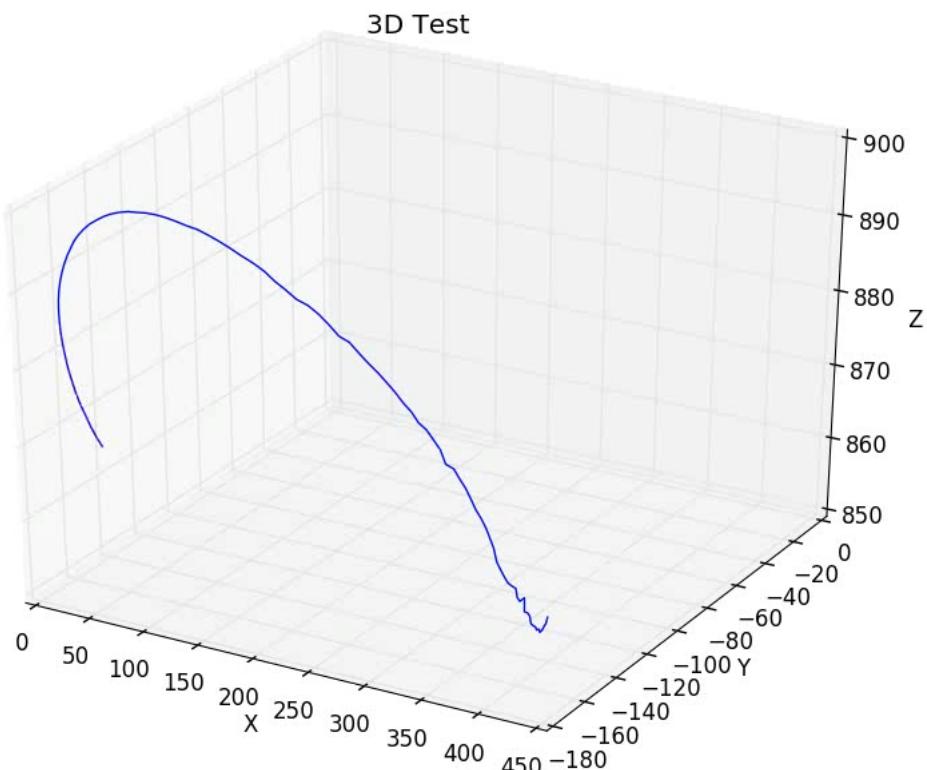


FIGURE 4 – Visualisation obtenue avec Python pour un capteur

Nous avons aussi écrit un code permettant de visualiser les trajectoires de plusieurs capteurs simultanément ainsi que l'ajout de bruit à une de ces trajectoires (Figure 5). Cela nous permettrait notamment d'obtenir une base de données de trajectoires plus importante à partir des trajectoires que nous avons enregistrées avec la plateforme AMIGO.

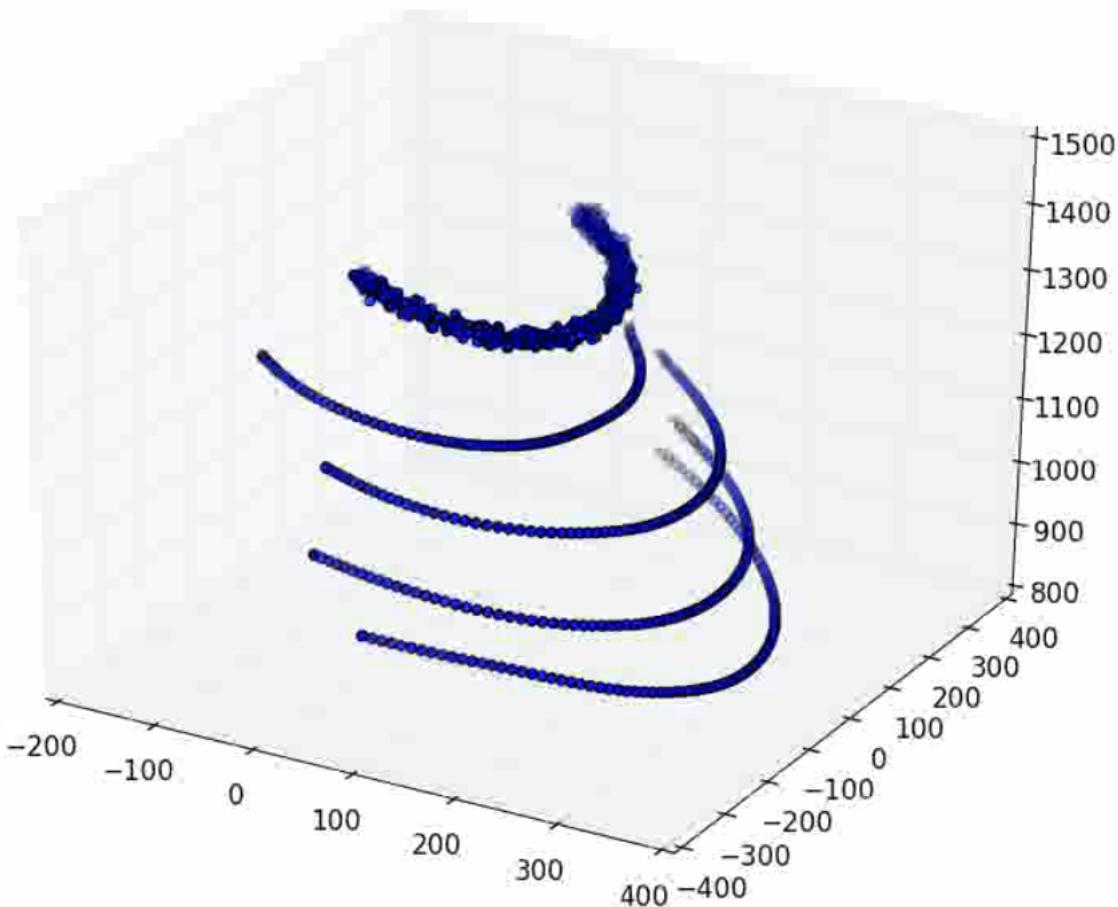


FIGURE 5 – Visualisation obtenue avec Python pour plusieurs capteurs avec ajout de bruit sur l'un d'eux

## 5.5 Blender

Sous Blender, nous avons travaillé le format .c3d qui est un format adapté pour les enregistrements de mouvements en 3D [6] et qui est un format que nous pouvons obtenir en exportant des données avec le logiciel Nexus.

Le PE a réalisé un tutoriel pour réaliser un squelette sous Blender. À partir de ce squelette, le logiciel permet d'ajouter des textures aux membres et donc de créer un véritable modèle humain en 3D. Pour le moment, nous avons seulement tenté de créer les liaisons entre les capteurs d'un bras capturé par les caméras de la plateforme (Figure 6 - Vidéo complète : <https://drive.google.com/open?id=0B2f8o0BqCEeBMkhjU1lmLU1Ja0E>). Nous souhaitons explorer davantage les fonctionnalités de ce logiciel pour trouver un modèle de visualisation adapté à la prédiction de mouvement.

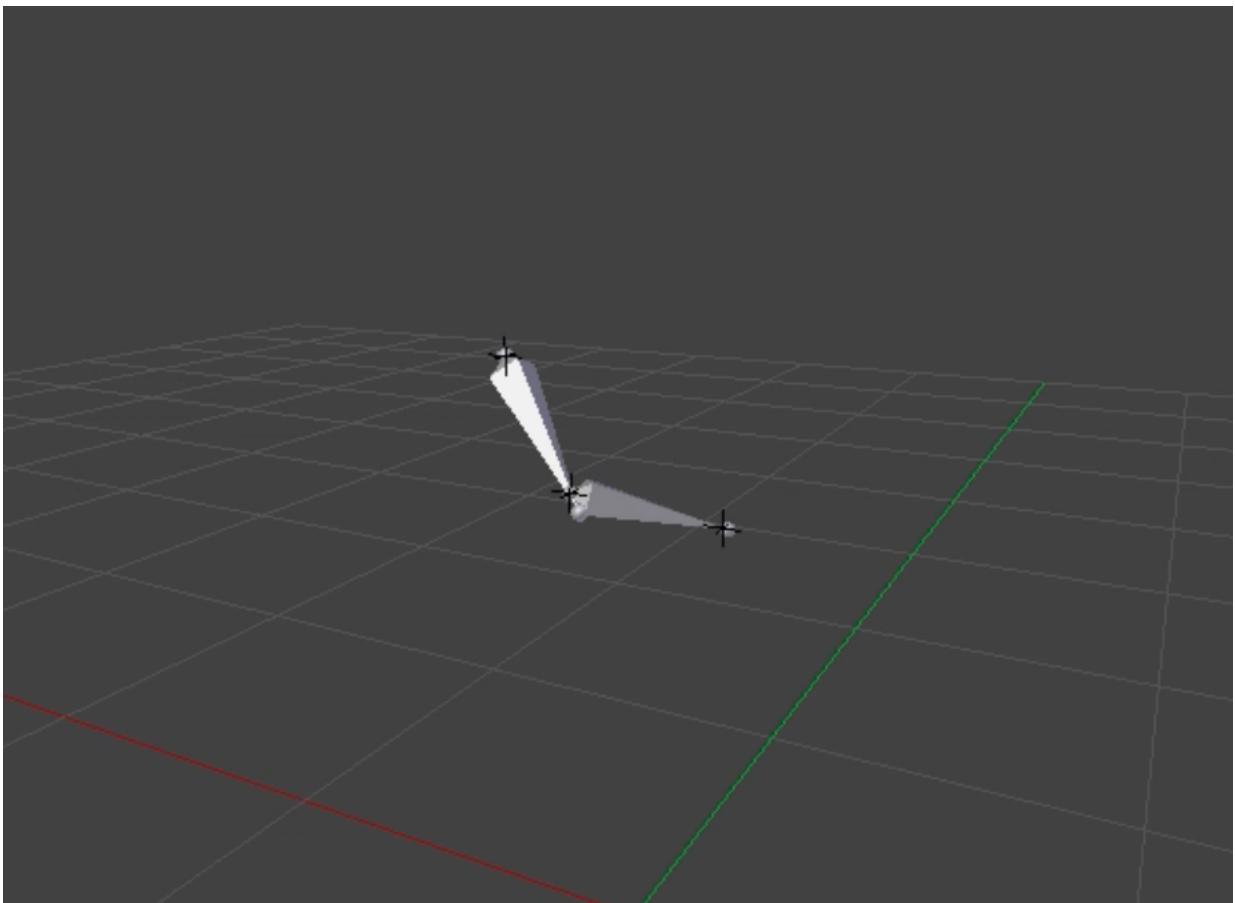


FIGURE 6 – Capture d’écran de la visualisation réalisée sur Blender

## 6 Études des modèles sur les trajectoires 3D

Afin d’effectuer le traitement des données 3D obtenues avec la plateforme AMIGO ou bien de données trouvées sur des bases de données en ligne, nous nous sommes intéressés aux travaux déjà réalisés pour la prédiction de mouvement.

### 6.1 Reconnaissance de formes

En ayant les données de traitement, on estime pouvoir appliquer des techniques de prédiction basées sur des classes de trajectoires.

Dans la littérature étudiée [7, 8, 9] ces méthodes sont appliquées pour les prédictions en 2D, mais on croit qu’il serait possible de les appliquer aussi en 3D. Pour une région donnée de l’espace, les objets en mouvement tendent à suivre des modèles typiques de mouvements qui dépendent de la nature de l’objet et la structure de l’environnement qui l’entoure. Ces techniques opèrent en deux phases :

1. **Phase d’apprentissage** : on observe les objets en mouvement dans l’espace afin de déterminer des modèles de mouvement typiques
2. **Phase de prédiction** : on utilise le modèle de mouvement appréhendé pour prédire les mouvements futurs de l’objet

Les techniques suivant ces deux façons d'opérer peuvent être classifiées en deux familles :

- **Techniques basées sur grilles** : elles sont dérivées du concept de grilles d'occupation. L'espace est modelé comme une grille et la phase d'apprentissage calcule la probabilité de transition de l'objet d'une cellule de la grille à une autre à côté.
- **Techniques basées sur les classes** : les trajectoires similaires observées partiellement ou complètement sont rassemblées en classes. Chaque classe a une trajectoire qui la représente et qui est utilisée à titre de comparaison.

Comme les techniques basées sur les classes permettent de prendre en compte non seulement la position actuelle de l'objet mais aussi ses positions précédentes, elles sont de loin les meilleures si on parle de prédiction à long terme.

## 6.2 Problème rencontré

Cette méthode est la plus utilisée pour la prédiction de mouvements humains. Néanmoins, nous n'avons pas réussi à comprendre comment faire pour déterminer les critères de similarité entre les trajectoires. De nombreux paramètres des trajectoires 3D peuvent influencer la ressemblance de différents enregistrements : le point de départ, la durée de l'enregistrement, l'échelle du mouvement etc. Nous ne savons pas comment prendre en compte cela dans des études de similarité sur les positions, les vitesses ou d'autres paramètres auxquels nous pouvons avoir accès à partir des données expérimentales. De plus, les études trouvées sur la prédiction de mouvement ne donnaient souvent qu'un aspect très théorique des critères pour créer les classes de mouvements et ce n'est pas à notre portée. Nous n'avons pas accès au code utilisé réellement. Il est alors difficile d'appliquer la théorie à notre cas. Il nous a donc été nécessaire de simplifier le problème.

# 7 Simplification du problème : trajectoires 2D

Suite à notre incapacité à trouver des codes pertinents et utilisables pour la prédiction de mouvements 3D, notre tuteur nous a conseillé de simplifier le problème et de s'intéresser à de la prédiction de mouvements 2D dans un premier temps.

Nous nous sommes intéressés à une interface de prédiction de l'écriture en ligne [10]. Celle-ci permet de visualiser en temps réel les prédictions effectuées par l'algorithme en traçant les trajectoires les plus probables lors du tracé des lettres dans l'encadré prévu. Ce site met à disposition le code utilisé pour la prédiction ainsi qu'une partie des données utilisées [11]. L'étude de l'interface d'écriture grâce au répertoire GitHub nous montre que le code n'est pas écrit sous Python que nous avions utilisé précédemment mais sous Javascript. Ne maîtrisant pas cet langage, nous avons cherché d'autres codes similaires en Python. Le projet [12] nous a intéressé particulièrement et sera expliqué par la suite.

## 7.1 Principe général

Dans le projet [12] l'auteur d'abord explique comment il réalise la phase d'apprentissage évoquée précédemment (Section 6.1) à partir de la base de donnée MNIST (Mixed National Institute of Standards and Technology) [13]. Cette base de données est présentée dans la suite (Section 7.2). Le programme va, à partir de cette base de données, établir un modèle qui permet de reconnaître les chiffres présents sur les images de la base de données. Nous avons

tenté d'effectuer cette phase d'apprentissage sur nos ordinateurs à partir de la même base de données. Il s'est avéré que certains modules utilisés dans le code n'étaient pas compatibles avec toutes les versions de Python. Il a fallu passer plusieurs heures à régler toutes les erreurs liées à ces modules (en particulier le module TensorFlow [14]) qui n'avaient pas non plus été utilisés par les projets avec qui nous sommes en contact.

Le projet [12] a également comme but la reconnaissance des chiffres écrits à la main par un utilisateur quelconque en utilisant la même bibliothèque Python que précédemment : TensorFlow. Nous pouvons ainsi voir l'efficacité de la généralisation du modèle établi précédemment pour une application réelle.

## 7.2 Base de données MNIST

L'auteur utilise la base de données MNIST [15]. Elle consiste dans un groupe d'images de chiffres écrits à la main comme ceux de la figure 7.



FIGURE 7 – Exemples des images présents dans la base de données MNIST

Cette base contient aussi des étiquettes associées à chaque image, ce qui permet de savoir quel chiffre elle représente. À titre d'exemple, les étiquettes des images dans la figure 7 sont 5, 0, 4 et 1 respectivement.

Les données de la base sont divisées en deux parties :

1. 55000 données d'apprentissage
2. 10000 données de test

Avec cette division on peut être sûr que le modèle que l'on crée peut être utilisé pour de nouvelles images. Le modèle est créé avec les données d'apprentissage et testé avec les données de test.

## 7.3 Régression Softmax - Modèle mathématique

Chaque image de la base MNIST est un chiffre écrit à la main entre 0 et 9. Il n'y a donc que dix possibilités d'étiquette pour chaque image. Nous souhaitons que le modèle (appelé Régression Softmax) soit capable de déterminer les probabilités pour qu'une image soit liée à chaque étiquette. Par exemple, le modèle pourrait évaluer qu'une image d'un 9 soit à 80% un neuf, mais donner une possibilité de 5% pour qu'il soit un huit (grâce à la boucle supérieure commune au 9 et au 8 par exemple) et une faible probabilité d'être chacun des autres chiffres.

Ce cas représente le cas idéal de la régression Softmax : elle nous donne une liste de valeurs pour chaque étiquette entre 0 et 1, dont la somme vaut 1. Ces valeurs représentent, dans notre cas, les probabilités cherchées. La démarche est divisée en deux parties. Premièrement, on fait la somme des raisons qui font que l'image évaluée soit attachée à une certaine étiquette. Ensuite, les raisons sont converties en probabilités.

Pour trouver qu'une image donnée est attachée à une étiquette en particulier, le modèle effectue une comparaison des pixels en attribuant des poids à chacun des pixels de l'image.

Le poids à attribuer à chaque pixel est soit positif s'il valide la ressemblance avec le chiffre de l'étiquette ou négatif s'il invalide la ressemblance. En faisant une somme pondérée pour chacune des étiquettes, on obtient les probabilités. Un exemple est donné en figure 8.

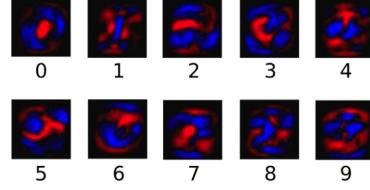


FIGURE 8 – Diagramme affichant les poids sur un exemple d'image. Le rouge représente le poids négatif, alors que le bleu représente le poids positif [16]

La régression prend aussi en compte un terme de partialité. Cela permet de prendre en compte le fait que certains résultats sont plus probables, pour n'importe quelle entrée. L'équation suivante permet de calculer les critères de ressemblances est la suivante :

$$X_i = \sum_j W_{i,j} x_j + b_i \quad (1)$$

où  $X_i$  est la valeur de la ressemblance à une étiquette  $i$ ,  $W_{i,j}$  est le poids attribué au pixel  $(i, j)$  et  $b_i$  le terme de partialité pour l'étiquette  $i$ .

Pour passer aux probabilités, il faut appliquer la fonction Softmax donnée par l'équation suivante :

$$y_i = softmax(X_i) = \text{normalise}(e^{X_i}) = \frac{e^{X_i}}{\sum_j e^{X_j}} \quad (2)$$

La régression Softmax est plus facilement visualisé à travers le diagramme dans la figure 9 ou sous forme matriciel dans la figure 10. Pour que la régression soit applicable à une trajectoire, il faudra que le modèle soit capable de traiter beaucoup plus d'entrées représentées par tous les points de l'espace dans lequel le mouvement a lieu.

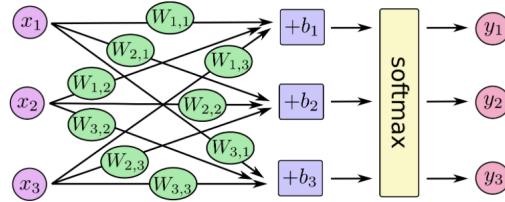


FIGURE 9 – Diagramme de fonctionnement de la régression [16]

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax} \left( \begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

FIGURE 10 – Régression représenté sous forme matriciel [16]

## 7.4 Implémentation Python du modèle

La bibliothèque TensorFlow est utilisée pour l'implémentation sous Python. La principale différence avec d'autres bibliothèques dédiées au calcul numérique comme NumPy est l'utilisation d'un graphe d'interactions de opérations traitées en dehors Python avec des codes hautement efficaces dans d'autres langages de programmation. NumPy, de son côté, réalisera les opérations une par une de façon indépendante.

Tout d'abord la base de données MNIST est importée ainsi que la bibliothèque TensorFlow.

```
1 import tensorflow as tf
2 from tensorflow.examples.tutorials.mnist import input_data
3 mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

Ensuite, les variables qui vont être utilisées pour la modélisation sont créées et le modèle est défini.

```
1 x = tf.placeholder(tf.float32, [None, 784])
2 W = tf.Variable(tf.zeros([784, 10]))
3 b = tf.Variable(tf.zeros([10]))
```

```
1 y = tf.nn.softmax(tf.matmul(x, W) + b)
```

Pour améliorer le modèle, il faut définir une méthode qui mesure son efficacité. Ceci est réalisé en mesurant le coût, ou la perte qui est représentée par la distance entre le modèle calculé et le modèle idéal. Le but ici est donc minimiser cette distance. C'est la fonction d'entropie croisée qui est utilisée pour cela. Elle est définie par :

$$H_{y'}(y) = - \sum_i y'_i \log(y_i) \quad (3)$$

Où  $y$  est la distribution calculée et  $y'$  la vraie distribution.

Il faut ensuite créer la variable qui recevra la vraie distribution et après définir la fonction d'entropie croisée.

```
1 y_ = tf.placeholder(tf.float32, [None, 10])
2 cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_*tf.log(y), reduction_indices=[1]))
```

Avec la fonction de perte ainsi définie, il faut dire au système géré par TensorFlow comment nous souhaitons qu'il modifie les variables et minimise cette fonction. C'est la rétro-propagation du système qui est utilisée. C'est-à-dire que nous choisissons l'algorithme du gradient avec un taux d'apprentissage de 0,5 pour la modification des variables. Cet algorithme modifie les variables à chaque fois vers la minimisation de la perte.

```
1 train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

Toutes les variables nécessaires sont définies, il faut maintenant les initialiser et définir l'outil d'enregistrement du modèle généré.

```
1 init = tf.global_variables_initializer()
2 saver = tf.train.Saver()
```

Le calcul peut alors s'exécuter dans une session de la plateforme **TensorFlow**. Il s'agit d'une boucle de 1000 itérations qui à chaque itération choisit 100 images aléatoirement. Le modèle "apprend" les étiquettes et ajuste les variables pour minimiser l'erreur. Cette méthode est appelée *entraînement stochastique*. Après finalisation de la boucle, le modèle est enregistré pour pouvoir l'utiliser en prédiction par la suite.

```

1 with tf.Session() as sess:
2     sess.run(init_op)
3     for i in range(1000):
4         batch_xs, batch_ys = mnist.train.next_batch(100)
5         sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
6
7     save_path = saver.save(sess, "model.ckpt")

```

Après avoir obtenu le modèle avec la méthode précédente, nous pouvons préparer la session pour réaliser le test. Il faut créer un vecteur pour comparer les résultats prédits avec les vrais. Ce vecteur est constitué de 0 et de 1 avec 0 si la prédiction est fausse et 1 si la prédiction est correcte. Le calcul de sa moyenne nous donne l'efficacité du modèle.

```

1 correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
2 accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
3
4 print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels
    }))

```

La précision obtenue pour le modèle créé est de 92%. Même si cela semble être déjà un bon résultat, il est en réalité possible d'atteindre des résultats jusqu'à 99,79% sur cette base de données [17]. Une direction à prendre déjà dans le projet sera l'amélioration de la précision de ces résultats.

## 7.5 Prédiction

Cette partie a comme but la reconnaissance d'un chiffre écrit à la main donné comme image lors de l'exécution du code. Le modèle créé précédemment est donc utilisé pour réaliser la prédiction du chiffre.

L'implémentation passe par deux fonctions. La première est responsable du traitement de l'image donnée pour qu'elle ait les mêmes propriétés que les images dans la base de données MNIST. Les propriétés requises sont :

1. L'image du chiffre doit s'ajuster à un carré de pixels de taille  $20 \times 20$  en préservant ses proportions.
2. L'image doit être centrée dans une carré de  $28 \times 28$  pixels.
3. Les pixels doivent être ordonnés par ligne en valant 0 (blanc) à 255 (noir).

La bibliothèque **PIL** (Python Imaging Library) est utilisée ici. Nous allons voir l'effet du programme sur une image pour laquelle il n'y pas de blanc entre le chiffre et l'extrémité de l'image. Un exemple est donné en Figure 11.

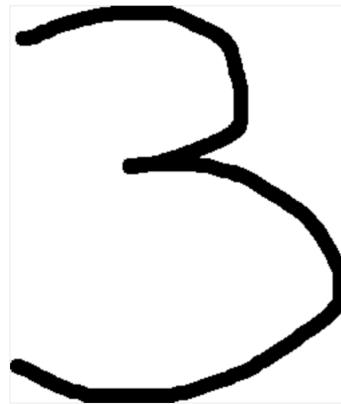


FIGURE 11 – Exemple d’image pour le traitement initial, la bordure grise définit la fin de l’image

Une démarche extensive est réalisée pour l’obtention des pixels prêts pour le comparaison avec le modèle. La démarche pas à pas est la suivante :

1. Chargement de l’image,
2. Conversion de l’image en noir et blanc,
3. Détermination de la dimension la plus grande de l’image,
4. Redimensionnement de l’image pour que la dimension la plus grande devienne 20 pixels et l’autre maintienne la proportion,
5. Amélioration de la qualité de l’image,
6. Création d’un canevas  $28 \times 28$ . Centrage de l’image à 4 pixels de l’extrémité à partir de la dimension la plus grande comme on peut voir en la Figure 12,
7. Enregistrer les valeurs des pixels de l’image dans le canevas  $28 \times 28$ ,
8. Normalisation des pixels.

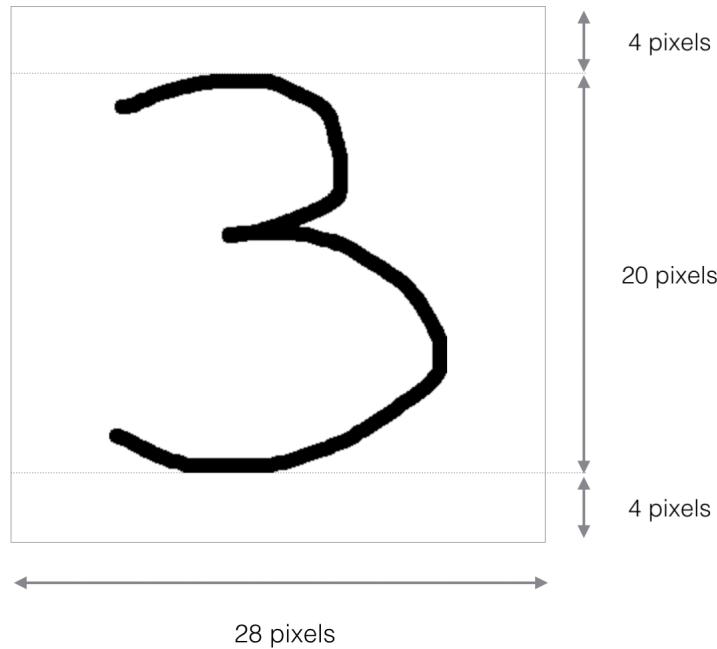


FIGURE 12 – Dimensionnement de l'image selon les propriétés de la base de données MNIST

La démarche décrite ci-dessus est exécutée par le code suivant :

```

1 def imageprepare(argv):
2     im = Image.open(argv).convert('L')
3     width = float(im.size[0])
4     height = float(im.size[1])
5     newImage = Image.new('L', (28, 28), (255))
6
7     if width > height:
8         nheight = int(round((20.0 / width * height), 0))
9         if (nheight == 0):
10             nheight = 1
11         img = im.resize((20, nheight), Image.ANTIALIAS).filter(ImageFilter.SHARPEN)
12         wtop = int(round(((28 - nheight) / 2), 0))
13         newImage.paste(img, (4, wtop))
14
15     else:
16         nwidth = int(round((20.0 / height * width), 0))
17         if (nwidth == 0):
18             nwidth = 1
19         img = im.resize((nwidth, 20), Image.ANTIALIAS).filter(ImageFilter.SHARPEN)
20         wleft = int(round(((28 - nwidth) / 2), 0))
21         newImage.paste(img, (wleft, 4))
22
23     tv = list(newImage.getdata())
24     tva = [ (255-x)*1.0/255.0 for x in tv]
25     return tva

```

Avec l'image préparée pour le test, les mêmes variables qu'avant sont définies. La différence sera alors dans l'exécution de la prédiction où il nous faut seulement restaurer le modèle enregistré. Ici nous avons rencontré un problème d'exécution lié à cette restauration qui nous a

bloqué. En n'arrivant pas à charger le modèle nous ne pouvons pas encore évaluer son efficacité.

```
1 def predictint(imvalue):
2
3     x = tf.placeholder(tf.float32, [None, 784])
4     W = tf.Variable(tf.zeros([784, 10]))
5     b = tf.Variable(tf.zeros([10]))
6     y = tf.nn.softmax(tf.matmul(x, W) + b)
7
8     init_op = tf.initialize_all_variables()
9     saver = tf.train.Saver()
10
11    with tf.Session() as sess:
12        sess.run(init_op)
13        saver.restore(sess, "model.ckpt")
14
15    prediction=tf.argmax(y,1)
16    return prediction.eval(feed_dict={x: [imvalue]}, session=sess)
```

## 8 Conclusion

Nous avons pris en main la plateforme AMIGO et le logiciel d'acquisition associé dans la première partie de ce projet. Cela nous a permis de réaliser plusieurs enregistrements du mouvement du bras droit et de réaliser l'exportation des données. Nous avons également travaillé en parallèle avec un autre groupe de PAr pour pouvoir surmonter certaines difficultés liées à l'acquisition d'un squelette plus complet et nous sommes maintenant en mesure de réaliser des acquisitions plus efficacement.

L'objectif initial était d'interpréter ces données afin de réaliser de la prédiction de mouvement sur le mouvement du bras droit en particulier. Néanmoins, suite à des difficultés rencontrées pour trouver une interprétation réalisable des données enregistrées avec la plateforme AMIGO ou bien de données 3D en général, nous nous sommes orientés vers une simplification du problème. Celle-ci consiste en considérant dans un premier temps l'espace 2D. Nous avons accès à des modèles de reconnaissance qui utilisent des images fixes. Ces modèles calculent des probabilité de similitude et cela pourrait être utilisé pour effectuer de la prédiction. Nous pourrons également voir par la suite si la prédiction de trajectoire peut être effectuée avec ces modèles en considérant une trajectoire comme une succession d'images.

Quant à l'avancée pour l'interface de visualisation qui constitue l'un des livrables, nous continuons de travailler sur des trajectoires enregistrées avec la plateforme pour le moment. Cela nous permet de développer des techniques générales de traitement des données avec différents logiciels comme Python et Blender.

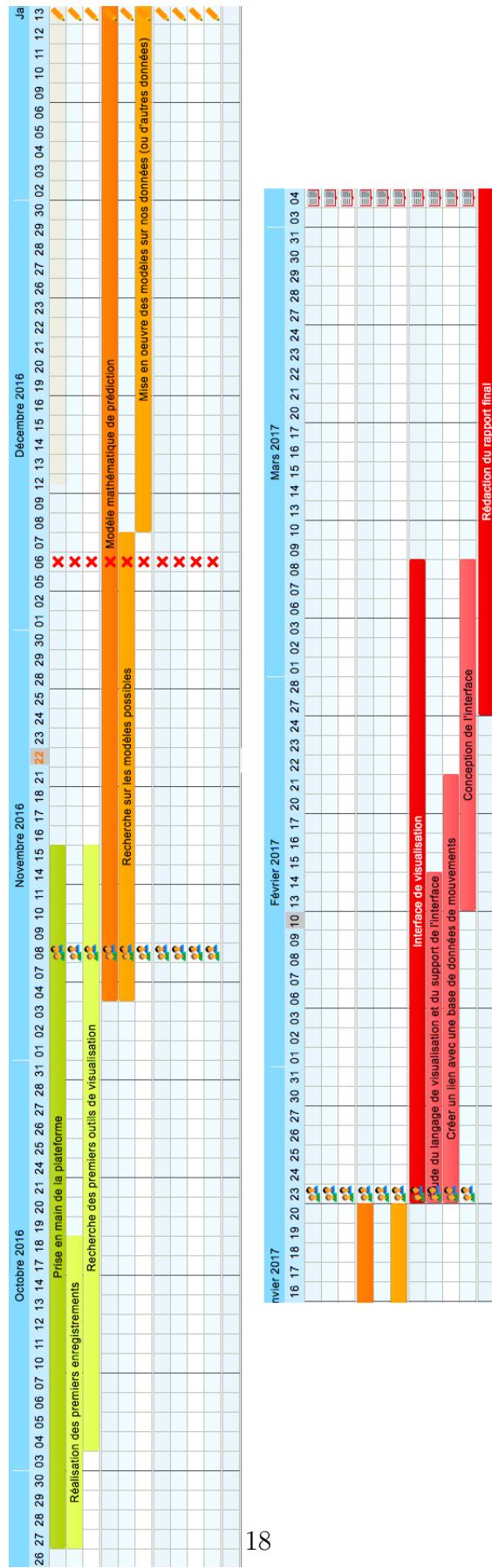
Dans la suite du projet, nous souhaitons pouvoir adapter le modèle de prédiction étudié aux trajectoires 2D. Il faudra alors travailler sur une visualisation 2D des trajectoires et des prédictions obtenues en fonction des probabilités de similarité associées. Celle-ci pourra probablement être adaptée des visualisations 3D sur lesquelles nous travaillons.

Nous souhaitons continuer le travail sur la visualisation 3D de plusieurs trajectoires en simultanée, notamment pour entamer les démarches qui permettraient la visualisation si nous trouvions des données de trajectoires et de trajectoires prédictes correspondantes.

## Références

- [1] Romain Vuillemot. Par 140 : Analyse et prédition de données spatio-temporelles sportives.
- [2] VICON. Vicon vero. <https://www.vicon.com/products/camera-systems/vero>.
- [3] VICON. Introducing nexus 2. <https://www.vicon.com/video/nexus-2-tutorials/introducing-nexus-2>.
- [4] Github. Amigocap. <https://github.com/AmigoCap>.
- [5] Blender. Open source 3d creation. <https://www.blender.org>.
- [6] Wikipedia. List of motion and gesture file formats. [https://en.wikipedia.org/wiki/List\\_of\\_motion\\_and\\_gesture\\_file\\_formats](https://en.wikipedia.org/wiki/List_of_motion_and_gesture_file_formats).
- [7] Dizan Alejandro Vasquez Govea and Thierry Fraichard. Motion prediction for moving objects : a statistical approach. *Proc. of the IEEE Int. Conf. on Robotics and Automation*, New Orleans, LA (US), 2004.
- [8] Arnaud Prouzeau, Anastasia Bezerianos, and Olivier Chapuis. Towards road traffic management with forecasting in wall displays. *Proceedings of the 2016 international conference on interactive surfaces and spaces*, Niagara Falls, Canada, 2016.
- [9] Thierry Artieres. Reconnaissance des formes. Techniques de l'ingénieur, octobre 2011.
- [10] Shan Carter, David Ha, Ian Johnson, and Chris Olah. Experiments in handwriting with a neural network. <http://distill.pub/2016/handwriting/>, 2016.
- [11] Utilisateur GitHub distillpub. post-handwriting. <https://github.com/distillpub/post--handwriting>.
- [12] Niek Temme. Use tensorflow to predict handwriting. <https://github.com/niektemme/tensorflow-mnist-predict/>.
- [13] Research Group on Computer Vision and Artificial Intelligence INF Universität Bern. Iam on-line handwriting database. <http://www.fki.inf.unibe.ch/databases/iam-on-line-handwriting-database>.
- [14] TensorFlow. An open-source software library for machine intelligence. <https://www.tensorflow.org>.
- [15] Yan LeCun, Corinna Cortes, and Chris Burges. MNIST handwritten digit database. <http://yann.lecun.com/exdb/mnist/>.
- [16] Google. MNIST for ML beginners. <https://www.tensorflow.org/tutorials/mnist/beginners/>.
- [17] Rodrigo Benenson. Classification datasets results. [http://rodrigob.github.io/are\\_we\\_there\\_yet/build/classification\\_datasets\\_results.html](http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html).
- [18] Utilisateur GitHub hardmaru. Generative handwriting using lstm mixture density network with tensorflow. <https://github.com/hardmaru/write-rnn-tensorflow>.

## Annexe A : Diagramme de Gantt





## Annexe B : Rapport d'activité

20

Dates	Activité	Liens	Notes
mardi 8 novembre	<b>RVP1</b>		
mercredi 9 novembre	<i>Restitution stage d'exécution</i>		
mardi 15 novembre			
mercredi 16 novembre			
mardi 22 novembre			
mercredi 23 novembre	<ul style="list-style-type: none"> <li>- Recherches bibliographiques sur les méthodes d'analyse des données de MoCap et finalisation de la visualisation des trajectoires enregistrées avec la plateforme AMIGO</li> <li>- Présentation de la plateforme Github avec Romain</li> </ul>	<p><a href="http://jakevdp.github.io/blog/2013/02/16/animating-the-lorentz-system-in-3d/">http://jakevdp.github.io/blog/2013/02/16/animating-the-lorentz-system-in-3d/</a> : code python pour la visualisation de données 3D, utilisation des fonctions set_3d_properties, set_data</p> <p><a href="https://github.com/AmigoCap/CaptureData">https://github.com/AmigoCap/CaptureData</a> : Plateforme github en commun avec le deuxième PAr travaillant sur la plateforme. Mise en commun du code python fourni par les deux groupes et accès facilité aux modifications mises en place chaque semaine. Le second PAr travaille majoritairement sur la visualisation des données alors que nous avons seulement établi un code de base pour faire un visualisation simple au premier abord. Nous pourrons nous servir de leurs avancées un peu plus tard.</p>	Beaucoup d'articles traitent d'analyse via machine learning. Certains articles traitent d'un sujet pratiquement identique à ce que l'on souhaite faire à partir des données de motion capture, mais il n'y a que très peu d'explications concernant l'algorithme utilisé. Les algorithmes ne sont pas accessibles dans la bibliographie. Les explications théoriques sont trop compliquées pour comprendre ce qu'ils font exactement à partir des données. Sur le site des techniques de l'ingénieur on a trouvé quelques articles sur la reconnaissance des formes qui nous donnent une base théorique plus solide sur le sujet.
mardi 29 novembre	<b>Rencontres avec l'autre groupe de PAr pour la réalisation d'un enregistrement sur squelette complet</b>	<p><a href="https://hal.archives-ouvertes.fr/hal-00656544/document">https://hal.archives-ouvertes.fr/hal-00656544/document</a> : Interprétation de Mouvements en Temps Réel - utilisation d'automates de mouvements pour reconnaître des mouvements et trouver des similitudes entre les mouvements pour créer des "cluster" qui regroupent des mouvements qui se ressemblent</p> <p><a href="https://hal.archives-ouvertes.fr/hal-01207938/document">https://hal.archives-ouvertes.fr/hal-01207938/document</a> : Reconnaissance d'actions humaines 3D par l'analyse de forme des trajectoires de mouvement - reconnaissance de mouvements similaires à l'aide de la méthode des k-plus-proches-voisins appliquée à des données d'une base publique</p>	Problèmes pour l'enregistrement du squelette complet. Nous avons réussi précédemment à enregistrer un squelette en trouvant l'étape manquante : il était nécessaire de créer des liens entre les segments constitués pour pouvoir effectuer la pipeline de reconnaissance du squelette. Nous n'avons pas réussi à refaire les liens entre les segments avec l'autre groupe. Il n'y a pas d'informations sur cette étape dans le livret d'aide dans la salle AMIGO et l'aide en ligne n'est pas suffisante.
mercredi 30 novembre	<b>Recherches sur la reconnaissance de formes</b>		
mardi 6 décembre	<i>Forum Perspectives</i>		
mercredi 7 décembre	<b>RDV avec Romain</b>	<p><a href="http://distill.pub/2016/handwriting/">http://distill.pub/2016/handwriting/</a> : site internet avec interface de visualisation d'une prédiction de l'écriture. À chaque instant lors du tracé du mot, on peut visualiser les trajectoires possibles prédictes par l'interface. Accès au code via github <a href="https://github.com/distillpub/post-handwriting">https://github.com/distillpub/post-handwriting</a></p> <p><a href="http://lstm.seas.harvard.edu">http://lstm.seas.harvard.edu</a> : cours de Harvard autour de LSTM</p> <p><a href="http://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Fragkiadaki_Recurrent_Network_Models_ICCV_2015_paper.pdf">http://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Fragkiadaki_Recurrent_Network_Models_ICCV_2015_paper.pdf</a> : utilisation des réseaux de neurones pour la reconnaissance et la prédiction de postures humaines</p>	<b>Simplifier le problème, les hypothèses</b> Prédiction de l'écriture -> Interface Pour chaque mouvement, arriver à chaque étape à indiquer quelles sont les prédictions Analyser, voir les mouvements clés qui comportent beaucoup d'information <b>LSTM Long Short Term Memory</b> Partir d'une trajectoire 2D, arriver à utiliser les mêmes outils que la reconnaissance d'écriture et peut-être reprendre la même démarche Modèle de validation : regarder si la prédiction correspond au jeu de validation <b>Réseau de neurones</b> Prise en compte de l'historique de manière plus simple Commencer plus simple avec un seul capteur ? Plus compliqué avec le motion capture. Présence d'un référentiel du mouvement lié au squelette Est-ce que le mouvement de la main seulement va être lié au mouvement du bras ? Formuler un problème similaire au modèle de l'écriture Formes géométriques avec un carré ? + Visualisation de comment le modèle s'exécute à chaque moment Et seulement après compliquer le modèle avec plusieurs capteurs et les dépendances entre eux Tensor flow : infrastructure de google pour faire du motion capture
mardi 13 décembre	<ul style="list-style-type: none"> <li>- Étude du code du site de prédiction de l'écriture</li> <li>- Recherche sur la Long Short Term Memory (LSTM) au niveau des mouvements</li> <li>- 2ème formation bibliographique</li> </ul>	<p>(<a href="https://github.com/hardmaru/write-rnn-tensorflow">https://github.com/hardmaru/write-rnn-tensorflow</a>) Ce lien a été utilisé pour le code python testé. Dans ce lien on trouve en plus des liens vers la base de données utilisée et le site web avec des articles de l'auteur de l'algorithme (Alex Graves)</p> <p><a href="http://colah.github.io/posts/2015-08-Understanding-LSTMs/">http://colah.github.io/posts/2015-08-Understanding-LSTMs/</a> : explications du principe de LSTM</p>	Le code python sur github sur l'écriture à main avec prédiction a été téléchargé avec sa base de données, mais il y a des problèmes au niveau de python sur l'ordinateur d'Arthur : problème avec Mac ? Un package ne semble pas se télécharger sous Anaconda

mercredi 14 décembre	<b>RDV avec Romain</b>		Romain nous a conseillé de tenir un journal de l'avancée du PE afin de noter les activités de chaque semaine. Nous y notons également la bibliographie trouvée à chaque séance afin de ne pas perdre les liens. Il faudrait aussi tenter de faire une première visualisation de trajectoires déviées à partir d'une trajectoire de base en ajoutant du bruit par exemple ou bien une composante aléatoire
mardi 3 janvier	- Fin de la rédaction de la chronologie - Création du code avec ajout de bruit aléatoire sur les trajectoires		Ajout de bruit aléatoire sur les trajectoires pour permettre d'augmenter le nombre de trajectoires dans notre base de données. Cela pourra permettre de regrouper plus facilement les trajectoires qui se ressemblent et de créer des clusters de trajectoires semblables comme ce qui est fait en deep learning.
mercredi 4 janvier			
mardi 10 janvier			
mercredi 11 janvier	- Rédaction du livrable pour le RVP 1 - Continuation de l'analyse du code Python du site de prédiction de l'écriture - Rendez-vous avec Romain pour discuter des objectifs		Afficher des trajectoires et des trajectoires bruitées autour avec plus ou moins de bruit Afficher à chaque instant les trajectoires les plus probables à chaque instant Ajouter une forme géométrique qui se déplace sur la trajectoire pour déterminer à chaque moment celles qui sont similaires - la trajectoire la plus proche ou un dégradé de couleur pour montrer les différentes trajectoires de plus en plus proches Utiliser Blender pour faire le rendu graphique
mardi 17 janvier			
mercredi 18 janvier			
lundi 23 janvier	<b>RVP 2 - 10h</b>		