Traitement et visualisation de données gestuelles captées par Optotrak

Travaux réalisés au laboratoire IDML (Université McGill)

Sous la responsabilité de Marcelo Wanderley

Juin – Août 2005

Charles VERRON
(contact : charles.verron@gmail.com)

1) Données et notations

Ce travail porte sur les mesures de geste de clarinettistes réalisées à McGill. Cependant la démarche décrite ici est applicable aux mesures faite à Amsterdam, et plus généralement à tout type d'instruments ou de données gestuelles capturées par Optotrak ou Vicon.

1) Description des mesures réalisées à McGill

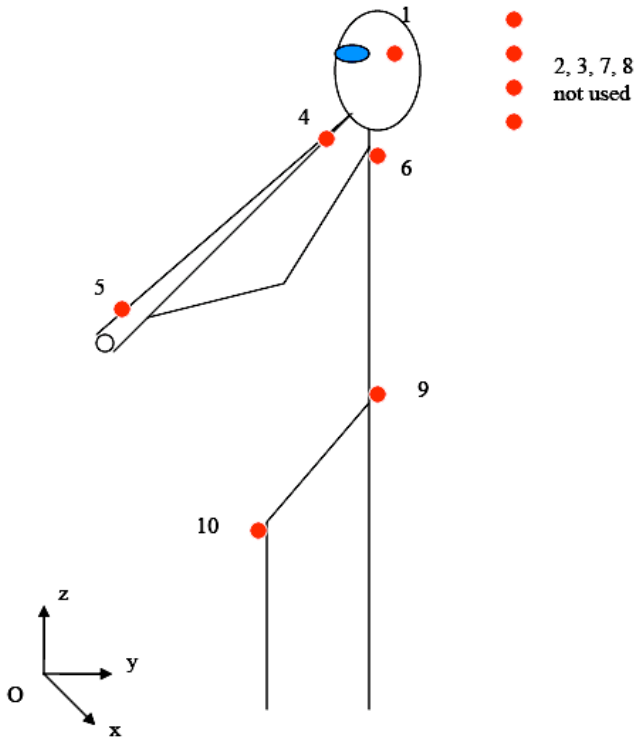
La description du protocole de mesure utilisé pour les données de clarinette de McGill se trouve dans le texte 'Optotrak Psych Lab Guide.doc'. Nous rappelons ici simplement la nature des données pour expliquer par la suite les conventions de notation qui on été choisies.

Sujets :
3 sujets (Lu, Mk et MJ). L'un des sujets a participé à deux séances de mesures à 9 mois d'intervalle (MJ1 et MJ2).

Morceau de musique :
La pièce jouée est un solo pour clarinette de Stravinsky.

Interpretations :
Chaque sujet a réalisé 7 interprétations de la pièce réparties selon 3 types : 2 Expressives, 4 Standards et 1 Immobile.

Markers d'optotrak :
1: head
2: NaN
3: NaN
4: barrel (near mouthpiece) of clarinet
5: bell of clarinet
6: shoulder
7: NaN
8: NaN
9: hip
10: knee



Pour chaque interprétation, on dispose des données suivantes :

- données de position des markers d'Optotrak dans un fichier ***.ndf***.
- prise vidéo de la performance dans un fichier ***.mov*** **(Attention : le .mov correspondant à la performance Standard 2 de MJ1 n'existe pas)**.
- son extrait de la vidéo (extension ***_v.wav***) (n'existe pas pour MJ1 Standard 2).
- son capté par l'Optotrak (synchrone avec les données de position des markers) contenu dans un fichier d'extension ***_au.ndf*** ou ***.aud***. Après extraction du fichier original, l'extension du fichier son obtenu est ***_o.wav***.

2) Données de position directement acquises par Optotrak

Pour identifier les données de position des markers d'Optotrak, nous avons choisi la convention suivante (nous donnons en exemple l'application sur Louise) :

- 2 lettres pour le nom du sujet

=> Lu

- 1 chiffre pour le numéro de la séance de mesure pour le sujet

=> Lu1

- 2 lettres pour le morceau de musique joué ('Sy' pour Stravinsky)

=> Lu1Sy

- 1 lettre pour le type d'interprétation (Expressif, Standard ou Immobile)

=> Lu1SyE, Lu1SyS, Lu1SyI

- 1 chiffre pour le numéro de l'interprétation au sein d'un type donné

=> Lu1SyE1, Lu1SyE2, Lu1SyS1, Lu1SyS2, Lu1SyS3, Lu1SyS4, Lu1SyI1

Les séances de mesure pour chaque sujet ont été enregistré dans un format propre à l'Optotrak (extension .ndf) est sont nommés avec cette convention : Lu1SyE1.ndf, Mk1SyS1.ndf, MJ1SyE2.ndf, MJ2SyI1.ndf…
Chacun de ces fichiers renferme l'information des 10 markers de l'Optotrak, sur trois coordonnées (x,y,z), soit 30 vecteurs de position. Après extraction du fichier, ces vecteur sont identifiés par :

- le numéro du marker (1..10)
- la coordonnée considérée (x, y ou z)
- une extension '_p' pour indiquer qu'il s'agit de données de position

Cela donne dans le cas de Louise :
Lu1SyE1_1x_p, Lu1SyE1_2x_p, Lu1SyE1_5z_p, Lu1SyS2_5z_p,…

3) Données calculées à partir des données de positions

Des données sont calculées à partir des vecteurs de positions.
Tout d'abord la vitesse et l'accélération. Ces nouveaux vecteurs sont nommées de la même manière mais avec l'extension '_v' pour la vitesse et '_a' pour l'accélération : **Lu1SyE1_1x_v, Lu1SyE1_1x_a,...**
La norme euclidienne ($\sqrt{x^2 + y^2 + z^2}$) est aussi calculée à partir des 3 dimensions (x,y,z) disponibles pour chaque grandeur évoquée précédemment. Pour identifier les vecteurs de norme,

nous remplaçons par ‘n’ la lettre indiquant la coordonnée dans les données. Cela donne dans le cas de Louise : **Lu1SyE1_1n_v, Lu1SyE1_1n_a,…**

Nous calculons aussi plusieurs angles :
- l’angle entre les markers 5, 4 et l’axe (-oz)
- l’angle entre les markers 5, 4 et 9
- l’angle entre les markers 4, 1 et (-oz),
- l’angle entre les markers 6, 9 et 10
- l’angle entre les markers 10, 9 et (-oz).

Note : Les deux derniers angles sont à considérer avec précaution car les markers 9 et 10 contiennent beaucoup de NaN (cf paragraphe : *Preprocessing*, p.7).

Nous calculons aussi les vitesses et accélérations angulaires correspondantes.
Les vecteurs obtenus sont nommés (pour Louise) : **Lu1SyE1_5_4_z_p, Lu1SyE1_5_4_9_v,…**

A Faire : il pourrait être intéressant de calculer l’angle entre les markers 5, 4, et le plan (Oyz) pour détecter les cercles du marker 5.

4) Données time warpées

Pour comparer différentes interprétations, il est nécessaire de synchroniser les données d’Optotrak correspondantes. Cette opération appelée ‘time warping’ est décrite plus loin dans ce document mais nous en parlons dès maintenant pour introduire les notations utilisées. Les données time warpées sont notées de la même manière que précédemment, mais avec ‘TW’ en plus : **Lu1SyE1TW_5z_p, Lu1SyS1TW_5_4_9_p,…**

5) Sauvegarde des données

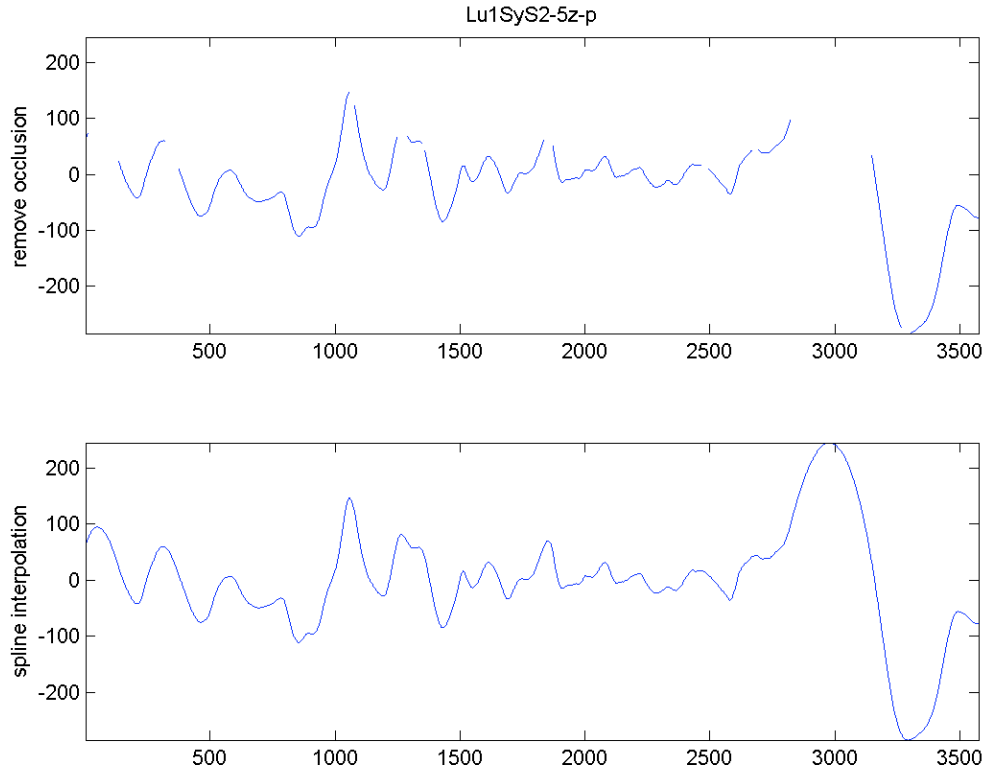
Les calculs sont longs pour traiter et time warper les données. Pour ne pas à avoir à les refaire à chaque fois, nous avons sauvegardé des fichier ***.mat*** qui contiennent les données déjà traitées et time warpées pour toutes les performances d’un sujet donné.
Ces fichiers sont nommés ainsi :
- identification du sujet et de la pièce de musique : **Lu1Sy**
- identification de l’interprétation qui a servi de modèle temporel pour le time warping des données : **_TW_Lu1SyE1**
- extention : **.mat**

Nous avons ainsi sauvegardé 12 fichiers comme **Mk1Sy_TW_Mk1SyS2.mat, MJ1Sy_TW_MJ1SyS4.mat,…** Le choix des données sauvegardées est expliqué dans la section de ce document dédiée au Time warping.

Notes importantes pour de nouvelles mesures : le fichier .ndf doit contenir 10 markers pour que toutes les fonctions Matlab de traitement des données fonctionnent correctement (car elles attendent des matrices de taille 30 qui correspondent aux 10 markers * 3 coordonnées).

2) Preprocessing

Pour obtenir les données décrites dans le chapitre précédent, il est nécessaire de traiter les données de position initialement présentes dans les fichiers NDF. Le principal problème vient des occlusions des markers de l'Optotrak qui se produisent assez fréquemment et se traduisent par des valeurs absurdes (de l'ordre de 10^{28}) dans les vecteurs de position issus des fichiers NDF. Ces valeurs aberrantes induisent des résultats encore plus absurdes dès lors que l'on calcule la vitesse et l'accélération.
Pour les éliminer, nous pratiquons un simple seuillage sur les données de position. Une interpolation par une fonction spline cubique par morceau permet ensuite de remplacer les points éliminés par des valeurs plus vraisemblables.



La figure ci-dessus illustre cette méthode d'interpolation pour le vecteur Lu1SyS2_5z_p. D'autres méthodes ont été envisagées (filtrage passe-bas, valeur constante assignée aux points manquants…) mais l'interpolation par spline semble la plus satisfaisante pour corriger les données de position.

Une fois les données interpolées, nous calculons vitesse, accélération et secousse (dérivée de l'accélération) par dérivation et filtrage passe-bas pour éliminer le bruit. C'est là que la spline cubique d'interpolation devient problématique.
Les données de position rajoutées par l'interpolation sont des polynômes d'ordre 3 sur toutes les plages temporelles ou les données des markers étaient NaN.
Lorsque l'on dérive sans filtrer, la vitesse devient un polynôme d'ordre 2 sur ces plages, l'accélération une fonction affine et la secousse une constante.

Le filtrage passe-bas que nous appliquons en même temps que la dérivation ne corrige en rien ce problème. Il permet seulement de diminuer le bruit dans les données réelles.

La seule chose à faire semble être d'avertir l'utilisateur lorsque les NaN sont trop nombreux dans les données de position pour un marker, et de lui conseiller de ne pas travailler sur ces données problématiques.

Nous donnons page 6 le nombre de NaN dans les différentes interprétations de 4 sujets, pour chaque marker.

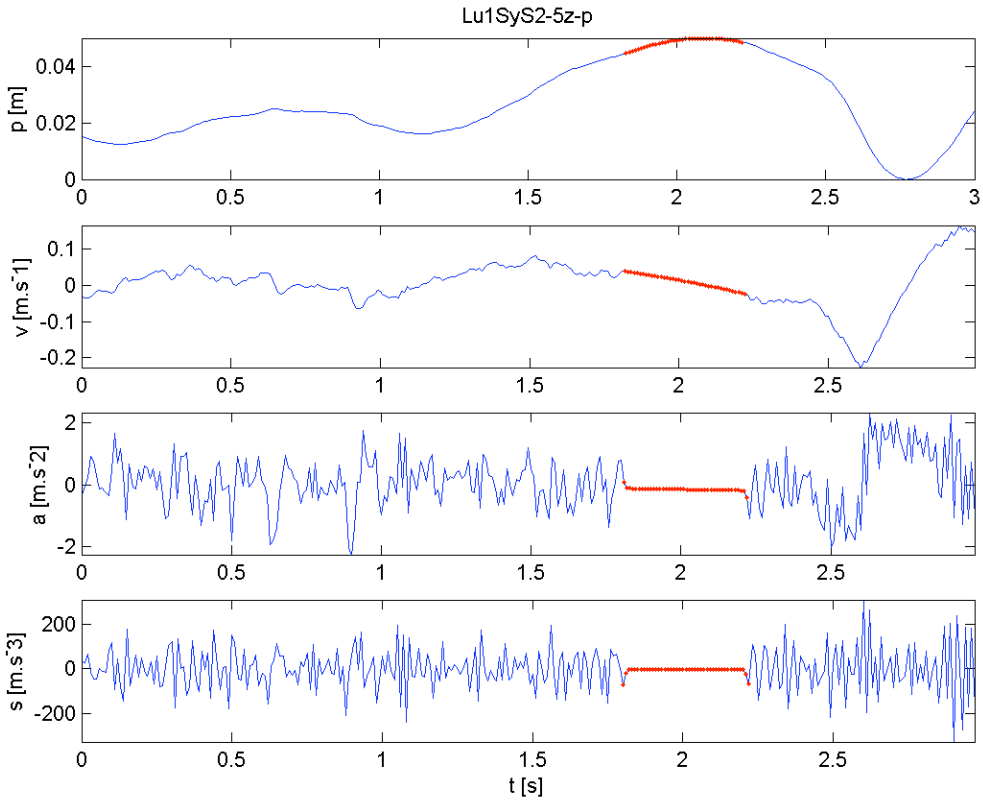
Les données sont colorées en bleu lorsqu'elles ne contiennent que des NaN. Cela signifie que ces markers n'étaient tout simplement pas utilisés lors de la campagne de mesure (sauf pour Lu1SyI1_10 qui est rempli de NaN alors qu'il était censé être utilisé, mais le marker a du tomber…).

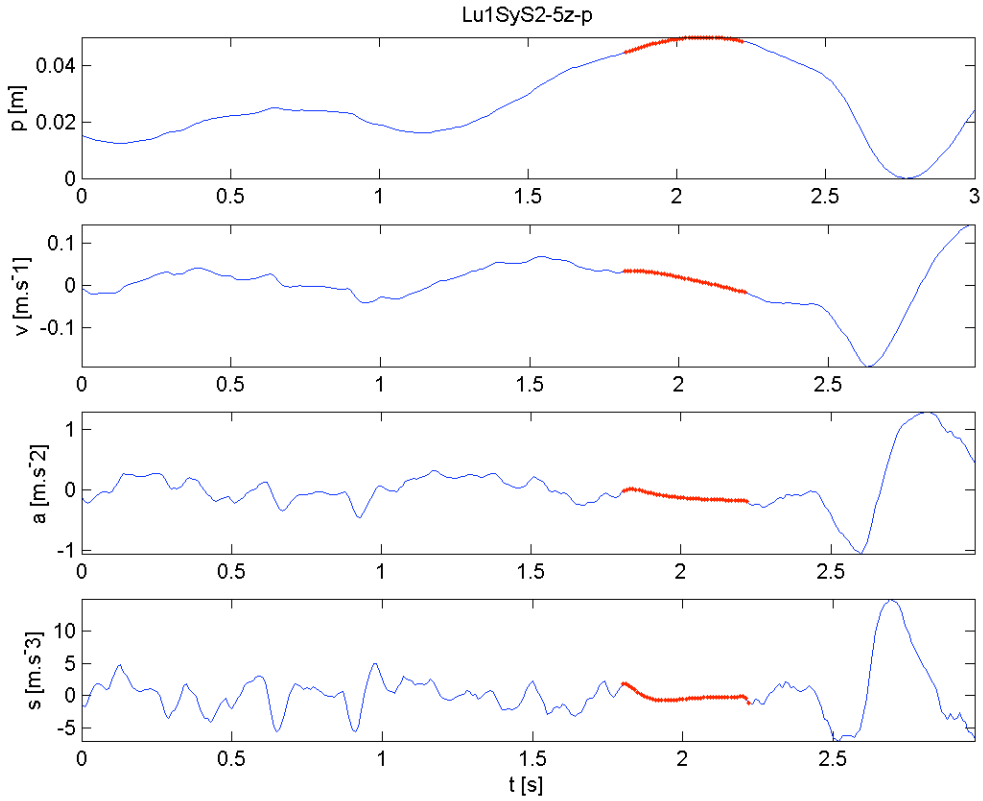
Les données sont colorées en rouge lorsqu'elles contiennent plus de 300 échantillons de NaN (3 secondes).
On voit qu'il s'agit presque toujours des markers 9 et 10, mais parfois aussi du marker 5 (Lu1SyS1_5: 17.58s, Lu1SyS2_5: 17.37s, Lu1SyS4_5: 3.35s) du marker 1 (MJ2SyS1_1: 30.19s, MJ2SyS3_1: 4.54s) et du marker 7 (Mk1SyS2_7: 3.18s).

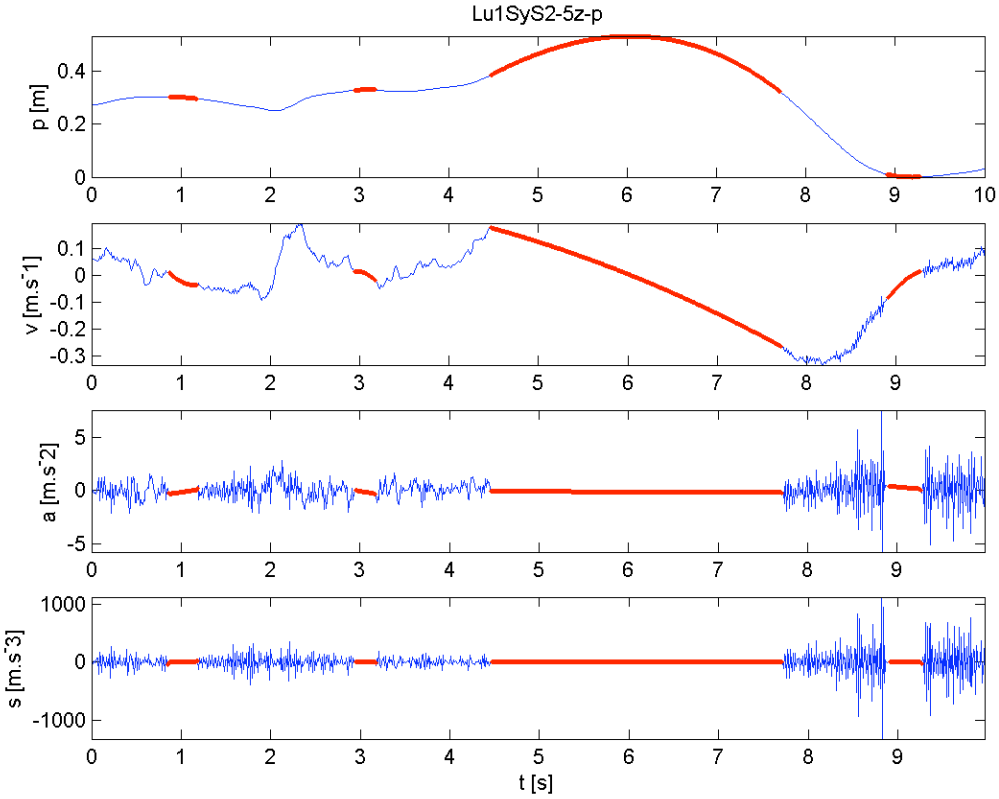
Les données en rouges sont donc à considérer avec la plus grande précaution.
Note : tous les markers de la performance Mk1SyI1 ont plus de 3 s de NaN car l'Optotrak n'a pas été arrêté à la fin de la mesure. Aussi les données sont valides, il faut simplement ne pas considérer toute la fin du vecteur (à partir de l'indice correspondant à la dernière note jouée).

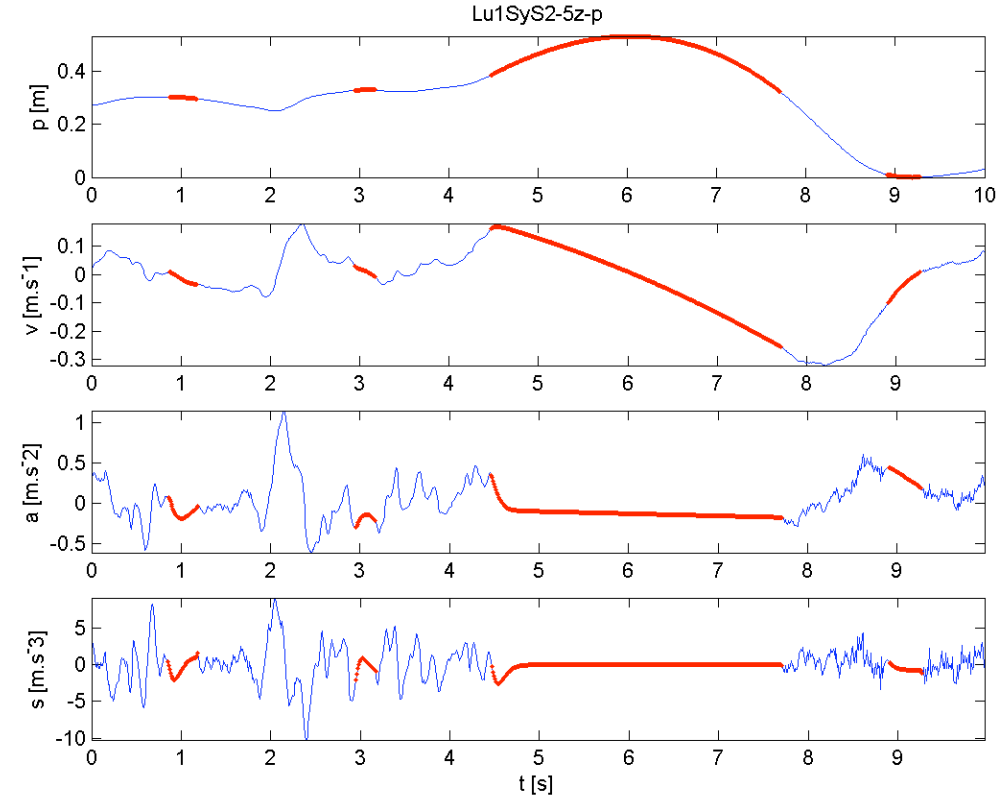
Les figures des pages 7 et 8 montre l'interpolation cubique pour deux extraits du vecteur de position Lu1SyS2_5z, particulièrement pourvus en NaN (représentés en rouge sur les graphiques). On y voit les conséquence sur la vitesse, l'accélération et la secousse, calculées sans (en haut) ou avec (en bas) filtrage passe-bas.

LOUISE		MARK		MarieJulie 1		Marie Julie 2	
'Lu1SyE1_8'	[7386]	'Mk1SyE1_8'	[8377]	'MJ1SyE1_8'	[9473]	'MJ2SyE1_8'	[7843]
'Lu1SyE1_3'	[7386]	'Mk1SyE1_3'	[8377]	'MJ1SyE1_3'	[9473]	'MJ2SyE1_3'	[7843]
'Lu1SyE1_2'	[7386]	'Mk1SyE1_2'	[8377]	'MJ1SyE1_2'	[9473]	'MJ2SyE1_2'	[7843]
'Lu1SyE1_10'	[3935]	'Mk1SyE1_9'	[3432]	'MJ1SyE1_4'	[207]	'MJ2SyE1_10'	[1188]
'Lu1SyE1_9'	[646]	'Mk1SyE1_10'	[536]	'MJ1SyE1_9'	[58]	'MJ2SyE1_9'	[587]
'Lu1SyE1_5'	[212]	'Mk1SyE1_7'	[174]	'MJ1SyE1_5'	[40]	'MJ2SyE1_7'	[0]
'Lu1SyE1_7'	[6]	'Mk1SyE1_1'	[7]	'MJ1SyE1_10'	[10]	'MJ2SyE1_6'	[0]
'Lu1SyE1_6'	[0]	'Mk1SyE1_6'	[0]	'MJ1SyE1_7'	[0]	'MJ2SyE1_5'	[0]
'Lu1SyE1_4'	[0]	'Mk1SyE1_5'	[0]	'MJ1SyE1_6'	[0]	'MJ2SyE1_4'	[0]
'Lu1SyE1_1'	[0]	'Mk1SyE1_4'	[0]	'MJ1SyE1_1'	[0]	'MJ2SyE1_1'	[0]
'Lu1SyE2_8'	[7690]	'Mk1SyE2_8'	[8247]	'MJ1SyE2_8'	[7485]	'MJ2SyE2_8'	[7977]
'Lu1SyE2_3'	[7690]	'Mk1SyE2_3'	[8247]	'MJ1SyE2_3'	[7485]	'MJ2SyE2_3'	[7977]
'Lu1SyE2_2'	[7690]	'Mk1SyE2_2'	[8247]	'MJ1SyE2_2'	[7485]	'MJ2SyE2_2'	[7977]
'Lu1SyE2_10'	[4037]	'Mk1SyE2_9'	[4135]	'MJ1SyE2_9'	[95]	'MJ2SyE2_10'	[1816]
'Lu1SyE2_9'	[507]	'Mk1SyE2_10'	[472]	'MJ1SyE2_10'	[0]	'MJ2SyE2_9'	[746]
'Lu1SyE2_1'	[162]	'Mk1SyE2_7'	[26]	'MJ1SyE2_7'	[0]	'MJ2SyE2_7'	[0]
'Lu1SyE2_7'	[114]	'Mk1SyE2_4'	[9]	'MJ1SyE2_6'	[0]	'MJ2SyE2_6'	[0]
'Lu1SyE2_5'	[85]	'Mk1SyE2_6'	[0]	'MJ1SyE2_5'	[0]	'MJ2SyE2_5'	[0]
'Lu1SyE2_6'	[0]	'Mk1SyE2_5'	[0]	'MJ1SyE2_4'	[0]	'MJ2SyE2_4'	[0]
'Lu1SyE2_4'	[0]	'Mk1SyE2_1'	[0]	'MJ1SyE2_1'	[0]	'MJ2SyE2_1'	[0]
'Lu1SyS1_8'	[7569]	'Mk1SyS1_8'	[8308]	'MJ1SyS1_8'	[7629]	'MJ2SyS1_8'	[9233]
'Lu1SyS1_3'	[7569]	'Mk1SyS1_3'	[8308]	'MJ1SyS1_3'	[7629]	'MJ2SyS1_3'	[9233]
'Lu1SyS1_2'	[7569]	'Mk1SyS1_2'	[8308]	'MJ1SyS1_2'	[7629]	'MJ2SyS1_2'	[9233]
'Lu1SyS1_10'	[3376]	'Mk1SyS1_10'	[1070]	'MJ1SyS1_9'	[1215]	'MJ2SyS1_1'	[3019]
'Lu1SyS1_5'	[1758]	'Mk1SyS1_9'	[123]	'MJ1SyS1_1'	[39]	'MJ2SyS1_10'	[2195]
'Lu1SyS1_9'	[355]	'Mk1SyS1_7'	[121]	'MJ1SyS1_10'	[0]	'MJ2SyS1_9'	[1281]
'Lu1SyS1_4'	[35]	'Mk1SyS1_4'	[83]	'MJ1SyS1_7'	[0]	'MJ2SyS1_7'	[0]
'Lu1SyS1_1'	[1]	'Mk1SyS1_6'	[0]	'MJ1SyS1_6'	[0]	'MJ2SyS1_6'	[0]
'Lu1SyS1_7'	[0]	'Mk1SyS1_5'	[0]	'MJ1SyS1_5'	[0]	'MJ2SyS1_5'	[0]
'Lu1SyS1_6'	[0]	'Mk1SyS1_1'	[0]	'MJ1SyS1_4'	[0]	'MJ2SyS1_4'	[0]
'Lu1SyS2_8'	[7574]	'Mk1SyS2_8'	[8484]	'MJ1SyS2_8'	[7642]	'MJ2SyS2_8'	[8944]
'Lu1SyS2_3'	[7574]	'Mk1SyS2_3'	[8484]	'MJ1SyS2_3'	[7642]	'MJ2SyS2_3'	[8944]
'Lu1SyS2_2'	[7574]	'Mk1SyS2_2'	[8484]	'MJ1SyS2_2'	[7642]	'MJ2SyS2_2'	[8944]
'Lu1SyS2_10'	[2107]	'Mk1SyS2_9'	[8315]	'MJ1SyS2_9'	[2202]	'MJ2SyS2_10'	[2751]
'Lu1SyS2_5'	[1737]	'Mk1SyS2_10'	[1640]	'MJ1SyS2_10'	[53]	'MJ2SyS2_9'	[706]
'Lu1SyS2_9'	[391]	'Mk1SyS2_7'	[318]	'MJ1SyS2_7'	[0]	'MJ2SyS2_1'	[2]
'Lu1SyS2_1'	[161]	'Mk1SyS2_1'	[25]	'MJ1SyS2_6'	[0]	'MJ2SyS2_7'	[0]
'Lu1SyS2_4'	[107]	'Mk1SyS2_6'	[0]	'MJ1SyS2_5'	[0]	'MJ2SyS2_6'	[0]
'Lu1SyS2_7'	[0]	'Mk1SyS2_5'	[0]	'MJ1SyS2_4'	[0]	'MJ2SyS2_5'	[0]
'Lu1SyS2_6'	[0]	'Mk1SyS2_4'	[0]	'MJ1SyS2_1'	[0]	'MJ2SyS2_4'	[0]
'Lu1SyS3_8'	[7366]	'Mk1SyS3_8'	[8348]	'MJ1SyS3_8'	[7424]	'MJ2SyS3_8'	[8441]
'Lu1SyS3_3'	[7366]	'Mk1SyS3_3'	[8348]	'MJ1SyS3_3'	[7424]	'MJ2SyS3_3'	[8441]
'Lu1SyS3_2'	[7366]	'Mk1SyS3_2'	[8348]	'MJ1SyS3_2'	[7424]	'MJ2SyS3_2'	[8441]
'Lu1SyS3_10'	[4067]	'Mk1SyS3_10'	[1834]	'MJ1SyS3_9'	[111]	'MJ2SyS3_10'	[2686]
'Lu1SyS3_9'	[577]	'Mk1SyS3_9'	[959]	'MJ1SyS3_10'	[0]	'MJ2SyS3_9'	[707]
'Lu1SyS3_5'	[100]	'Mk1SyS3_1'	[43]	'MJ1SyS3_7'	[0]	'MJ2SyS3_1'	[454]
'Lu1SyS3_1'	[91]	'Mk1SyS3_7'	[0]	'MJ1SyS3_6'	[0]	'MJ2SyS3_7'	[0]
'Lu1SyS3_7'	[0]	'Mk1SyS3_6'	[0]	'MJ1SyS3_5'	[0]	'MJ2SyS3_6'	[0]
'Lu1SyS3_6'	[0]	'Mk1SyS3_5'	[0]	'MJ1SyS3_4'	[0]	'MJ2SyS3_5'	[0]
'Lu1SyS3_4'	[0]	'Mk1SyS3_4'	[0]	'MJ1SyS3_1'	[0]	'MJ2SyS3_4'	[0]
'Lu1SyS4_8'	[7461]	'Mk1SyS4_8'	[7760]	'MJ1SyS4_8'	[7530]	'MJ2SyS4_8'	[7953]
'Lu1SyS4_3'	[7461]	'Mk1SyS4_3'	[7760]	'MJ1SyS4_3'	[7530]	'MJ2SyS4_3'	[7953]
'Lu1SyS4_2'	[7461]	'Mk1SyS4_2'	[7760]	'MJ1SyS4_2'	[7530]	'MJ2SyS4_2'	[7953]
'Lu1SyS4_10'	[5599]	'Mk1SyS4_9'	[2531]	'MJ1SyS4_9'	[77]	'MJ2SyS4_10'	[7105]
'Lu1SyS4_9'	[762]	'Mk1SyS4_10'	[1698]	'MJ1SyS4_10'	[0]	'MJ2SyS4_9'	[788]
'Lu1SyS4_5'	[335]	'Mk1SyS4_7'	[122]	'MJ1SyS4_7'	[0]	'MJ2SyS4_1'	[11]
'Lu1SyS4_1'	[238]	'Mk1SyS4_1'	[98]	'MJ1SyS4_6'	[0]	'MJ2SyS4_7'	[0]
'Lu1SyS4_4'	[91]	'Mk1SyS4_5'	[35]	'MJ1SyS4_5'	[0]	'MJ2SyS4_6'	[0]
'Lu1SyS4_7'	[18]	'Mk1SyS4_6'	[0]	'MJ1SyS4_4'	[0]	'MJ2SyS4_5'	[0]
'Lu1SyS4_6'	[0]	'Mk1SyS4_4'	[0]	'MJ1SyS4_1'	[0]	'MJ2SyS4_4'	[0]
'Lu1SyI1_10'	[7561]	'Mk1SyI1_8'	[18000]	'MJ1SyI1_8'	[7531]	'MJ2SyI1_8'	[7532]
'Lu1SyI1_8'	[7561]	'Mk1SyI1_3'	[18000]	'MJ1SyI1_3'	[7531]	'MJ2SyI1_3'	[7532]
'Lu1SyI1_3'	[7561]	'Mk1SyI1_2'	[18000]	'MJ1SyI1_2'	[7531]	'MJ2SyI1_2'	[7532]
'Lu1SyI1_2'	[7561]	'Mk1SyI1_10'	[10955]	'MJ1SyI1_9'	[126]	'MJ2SyI1_10'	[2916]
'Lu1SyI1_1'	[120]	'Mk1SyI1_7'	[9650]	'MJ1SyI1_10'	[0]	'MJ2SyI1_9'	[190]
'Lu1SyI1_5'	[29]	'Mk1SyI1_9'	[8760]	'MJ1SyI1_7'	[0]	'MJ2SyI1_1'	[5]
'Lu1SyI1_9'	[0]	'Mk1SyI1_5'	[8356]	'MJ1SyI1_6'	[0]	'MJ2SyI1_7'	[0]
'Lu1SyI1_7'	[0]	'Mk1SyI1_1'	[8026]	'MJ1SyI1_5'	[0]	'MJ2SyI1_6'	[0]
'Lu1SyI1_6'	[0]	'Mk1SyI1_4'	[2276]	'MJ1SyI1_4'	[0]	'MJ2SyI1_5'	[0]
'Lu1SyI1_4'	[0]	'Mk1SyI1_6'	[2063]	'MJ1SyI1_1'	[0]	'MJ2SyI1_4'	[0]

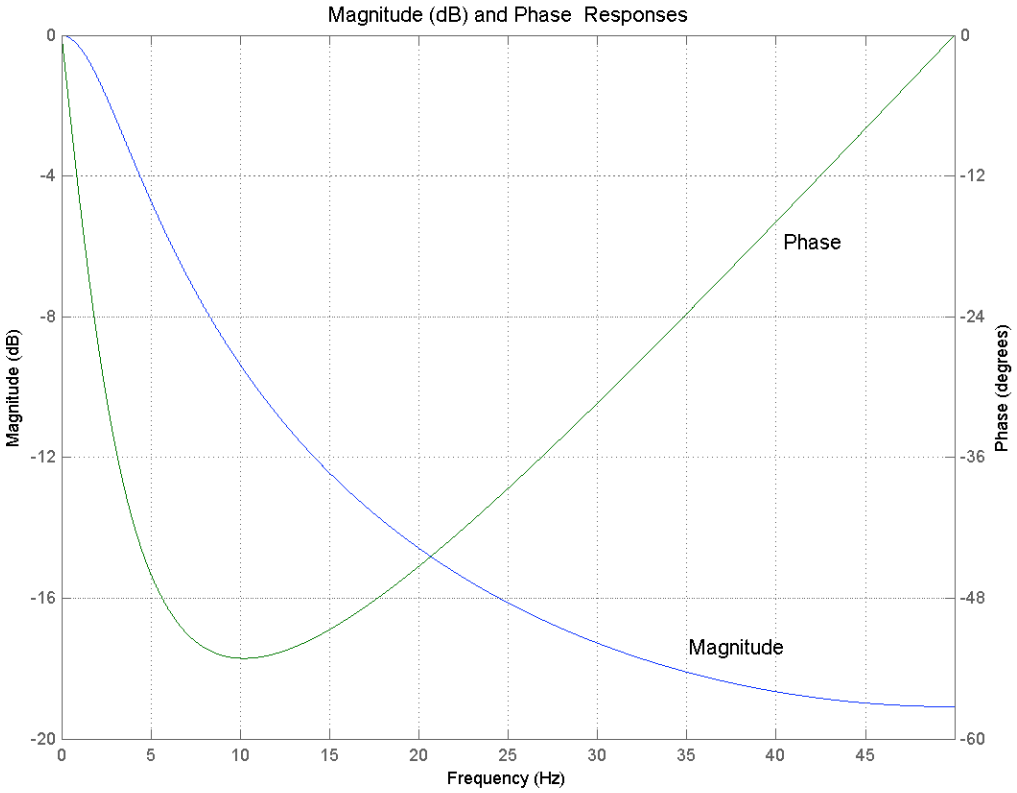

Lu1SyS2-5z-p
p [m]
v [m.s^-1]
a [m.s^-2]
s [m.s^-3]
t [s]


Lu1SyS2-5z-p
p [m]
v [m.s^-1]
a [m.s^-2]
s [m.s^-3]
t [s]


Lu1SyS2-5z-p
p [m]
v [m.s^1]
a [m.s^2]
s [m.s^3]
t [s]


Lu1SyS2-5z-p
p [m]
v [m.s^1]
a [m.s^2]
s [m.s^3]
t [s]

Nous donnons ici le diagramme de gain et de phase du filtre passe-bas utilisé. C'est un filtre du premier ordre (y(n) = 0.8 y(n-1) + 0.2 x(n)) de fréquence de coupure 3,5 Hz.



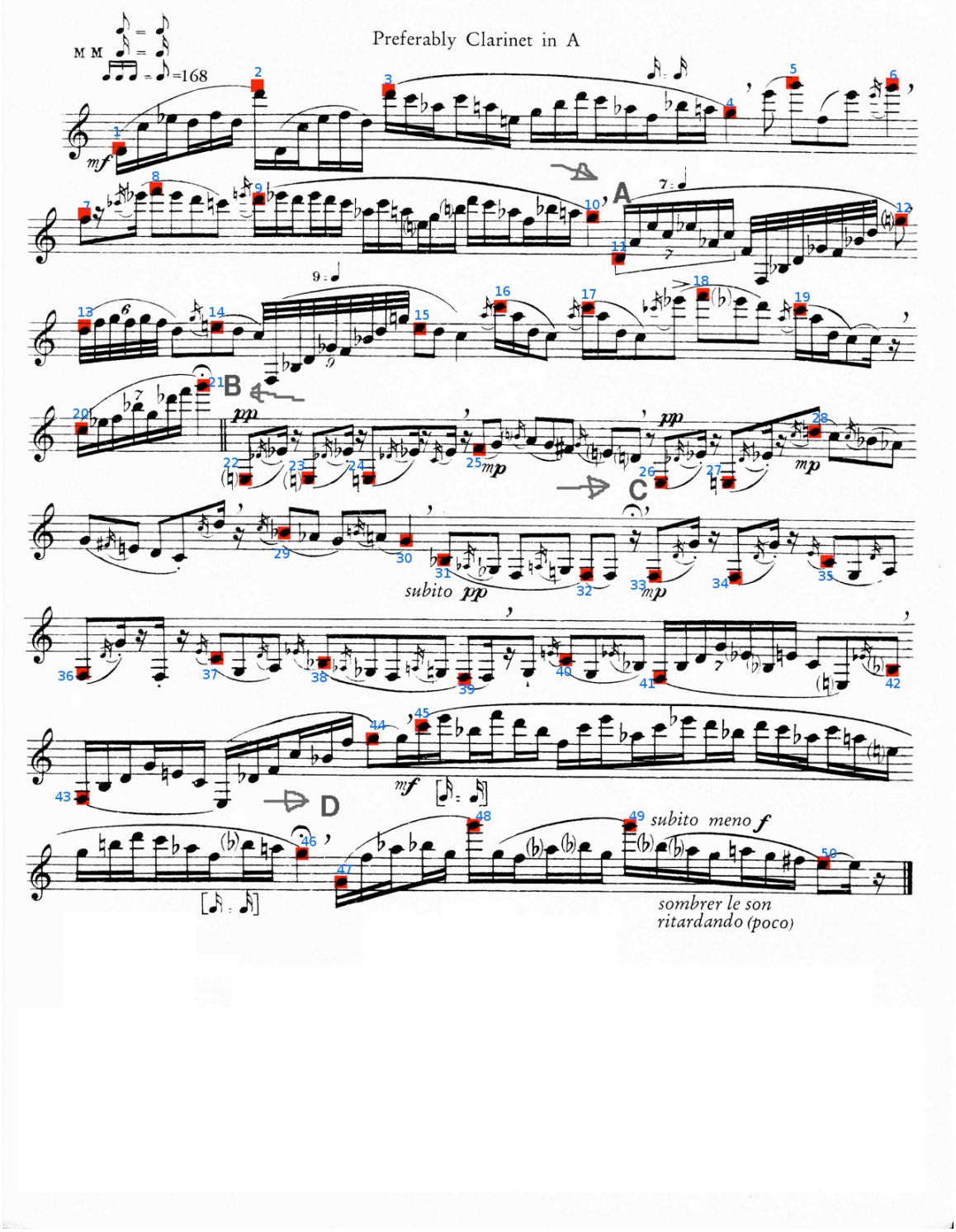
3) Time Warping

Pour pouvoir comparer différentes interprétations d'une même pièce, il est nécessaire de les 'synchroniser' : les instants correspondant aux mêmes notes dans la partition doivent se confondre sur le graphe. Cette déformation des données est appelée time warping (conformation temporelle en français).

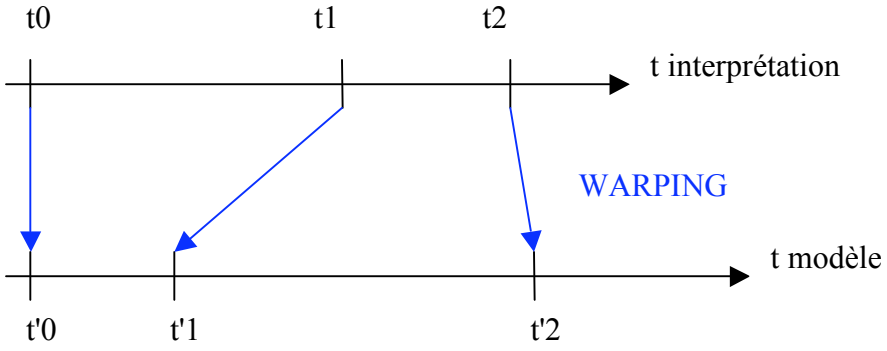
1) Méthode

Prenons une interprétation de la pièce, la première étape du time warping consiste à repérer les temps qui correspondent aux notes de la partition dans les données de l'Optotrak. Ce travail est réalisée 'à l'oreille' avec l'aide du logiciel Praat : les temps des notes sont repérés dans le fichier audio synchrone avec les données gestuelles. Cette technique est 'fiable' et évite les erreurs que l'on rencontre en essayant de traduire directement le fichier son en Midi, et en faisant un suivi de partition. Par contre elle est fastidieuse… et il serait beaucoup trop long de repérer les temps pour toutes les notes de la partition.

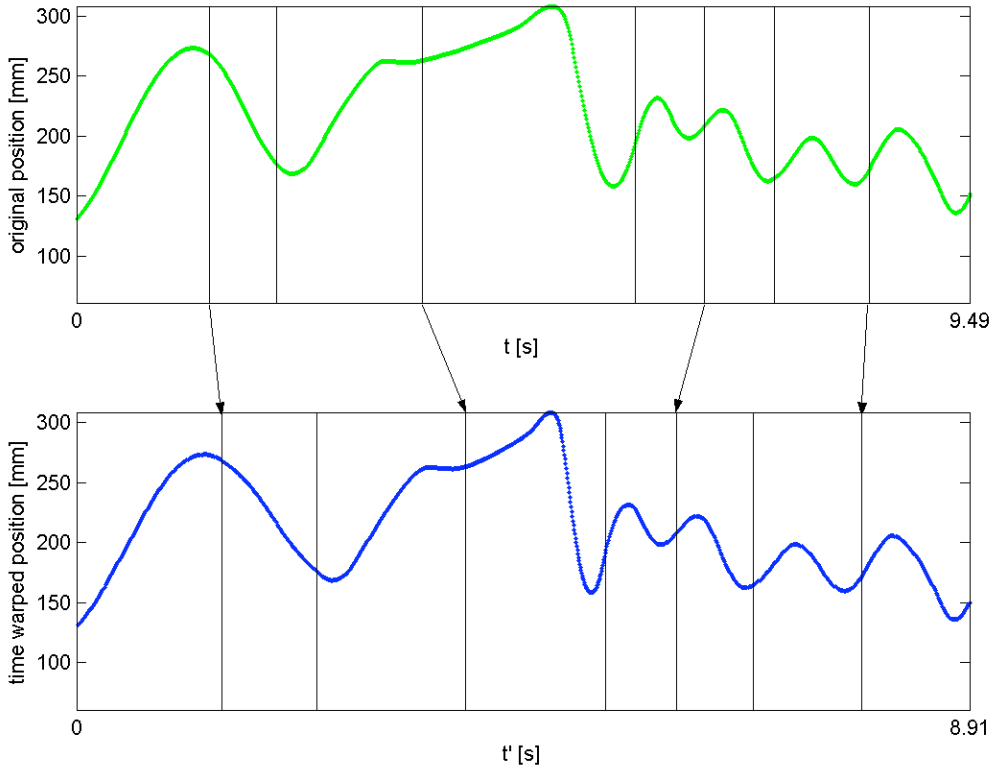
Une clarinettiste professionnelle a déterminé 50 notes dans la partition du solo de Stravinsky qui lui semblaient les plus adaptées pour délimiter les différentes phases du geste. Le repérage temporel a été fait pour ces 50 notes seulement.



Le même travaille de repérage est réalisé dans une autre interprétation (ou dans une partition MIDI de la pièce) qui servira de modèle temporel pour le time warping.
Notre time warping consiste à 'assimiler' les marqueurs temporels de l'interprétation à ceux du modèle correspondant aux mêmes notes. Nous donnons un exemple ici pour 3 notes fictives Do, Ré et Mi. Dans l'interprétation, Do est repéré à t0, Mi à t1 et Ré à t2. Dans l'interprétation modèle Do est repéré à t'0, Mi à t'1 et Ré à t'2.



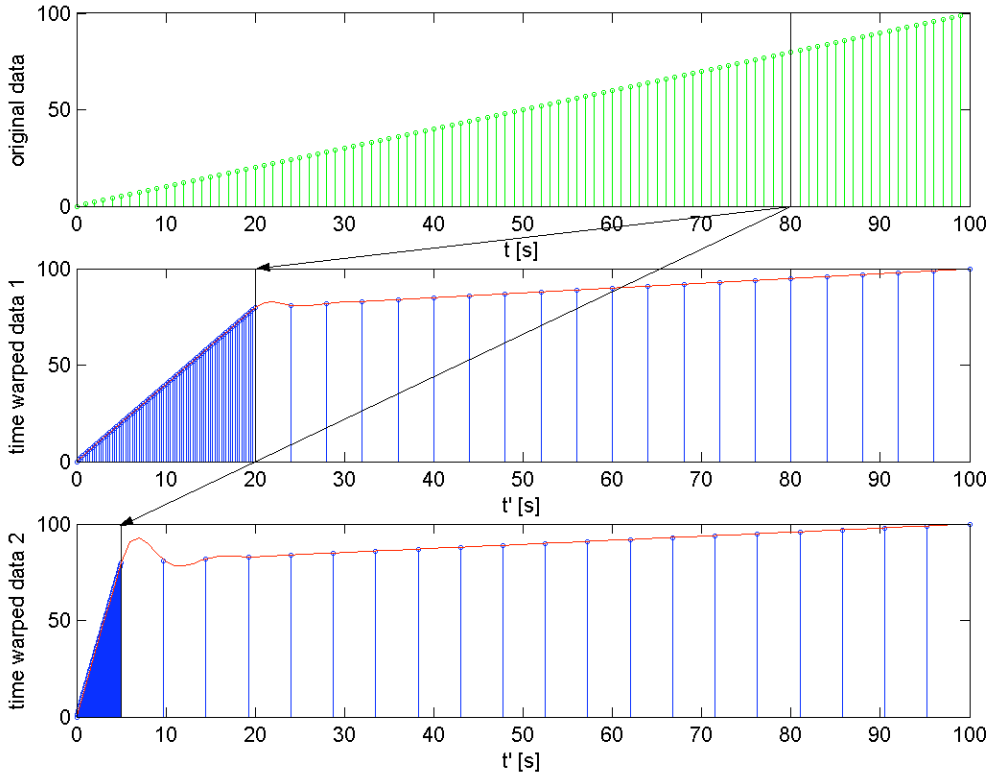
Entre les points de repère temporels t(i)..t(i+1), les données sont compressées ou dilatées linéairement. La figure suivante montre le résultat sur des données d'Optotrak réelle. Le vecteur est Lu1SyS2_5z_p et le modèle utilisé pour le repérage des temps est la partition MIDI du solo pour clarinette de Stravinsky.



Après l'opération de warping, les points ne sont plus régulièrement espacés dans le temps : l'échantillonnage est régulier par morceau, entre chaque marqueur temporel.
Un ré-échantillonnage des données est nécessaire. Nous utilisons encore pour cela une fonction spline cubique.

2) Précision

La précision de cette méthode de time warping reste à évaluer. Si l'écart entre les temps repérés dans l'interprétation à warper et ceux du modèle est trop grand, alors la fonction spline pourrait localement osciller. Un exemple est donné sur la figure ci-dessous :



Cependant cela semble ne pas poser de problème pour notre application, car on n'atteint jamais des facteurs de compression/dilatation très importants (par rapport aux fréquences du geste), dès lors qu'on ne prend pas un modèle aberrant (voir paragraphe suivant : *choix d'une interprétation modèle*).
Quant au repérage des temps, réalisé à la main dans Praat, on peut estimer que sa précision est de l'ordre de 1/10 de seconde, si l'utilisateur est vigilent.

L'expérience montre que les données time warpées avec cette méthode semblent correctes. Cependant, pour ne pas prendre de risque, tous les calculs (norme, angles...) sont fait sur les données originales et le résultat est ensuite time warpé. Cela évite de dégrader la précision. C'est plus coûteux en calcul (car plus de warping à faire) mais plus prudent dans un premier temps.

Notons tout de même cet exemple : pour Lu1SyS2_5n_v, on s'intéresse à la différence obtenue entre un calcul de norme sur les données originales puis time warpé, et un calcul de norme sur les données déjà time warpées (sur le même modèle temporel). Le maximum de cette différence est 3/1000 de la valeur moyenne de la norme. Donc le fait de time warper avant ou après le calcul de norme semble avoir des conséquences négligeables, ce qui est bien ce qu'on espérait. Nous n'avons pas encore trouvé le moyen de prouver cela mathématiquement.

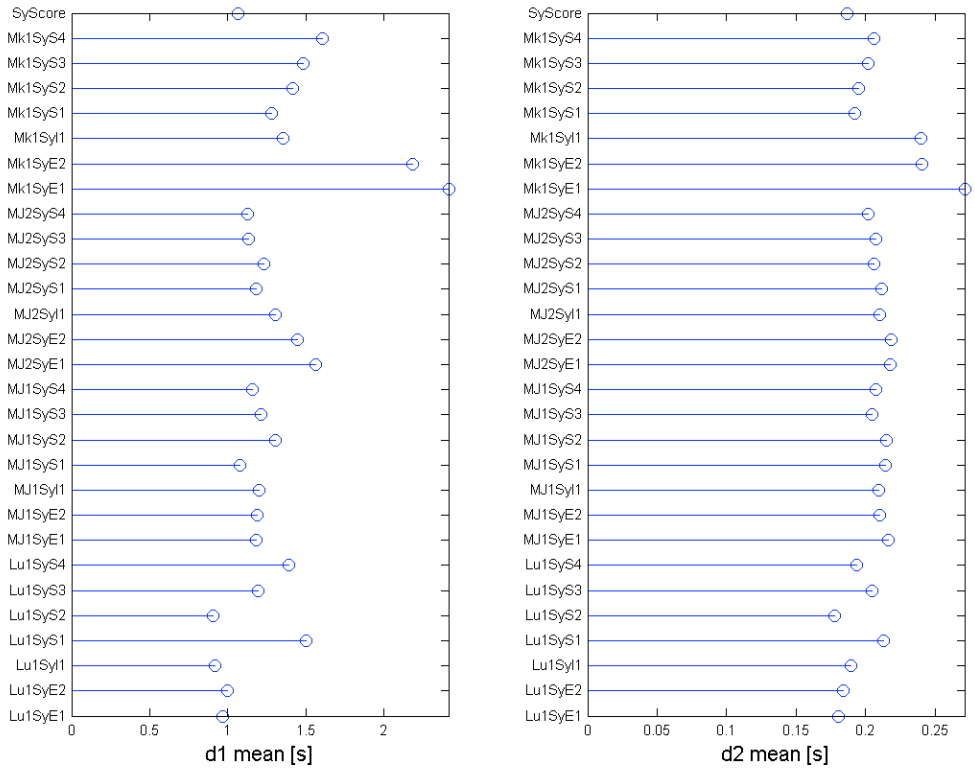
3) Choix d'une interprétation modèle

Pour pouvoir comparer toutes les performances de tous les sujets, il faut leur trouver un model temporel commun, qui 'déforme' chaque performance le moins possible. Nous proposons ici un calcul pour évaluer cette déformation.
Nous considérons qu'un bon indicateur de la 'déformation' infligée lors d'un time warping correspond au maximum de la différence de temps entre les segments temporels du model et ceux de l'interprétation à time warper.
Notons d_1 cet indicateur de 'déformation temporelle', t le vecteur des temps marqués dans l'interprétation et t' le vecteur des temps correspondants aux même notes dans l'interprétation modèle.

On a $$d_1 = \max_i |(t_{i+1} - t_i) - (t'_{i+1} - t'_i)|$$

Notre indicateur d_1 n'est rien d'autre qu'un calcul d'∞-distance entre les 2 vecteurs $(t_{i+1} - t_i)$ et $(t'_{i+1} - t'_i)$.
Il pourrait être intéressant de tester avec d'autres calculs de distance entre ces 2 vecteurs (euclidienne, p-distance) pour voir si ils correspondent mieux comme indicateur de déformation temporelle. Nous avons aussi fait le calcul avec la 1-distance (normalisée), ce qui donne un indicateur de 'déformation temporelle' d_2 :

$$d_2 = \underset{i}{mean} |(t_{i+1} - t_i) - (t'_{i+1} - t'_i)|$$

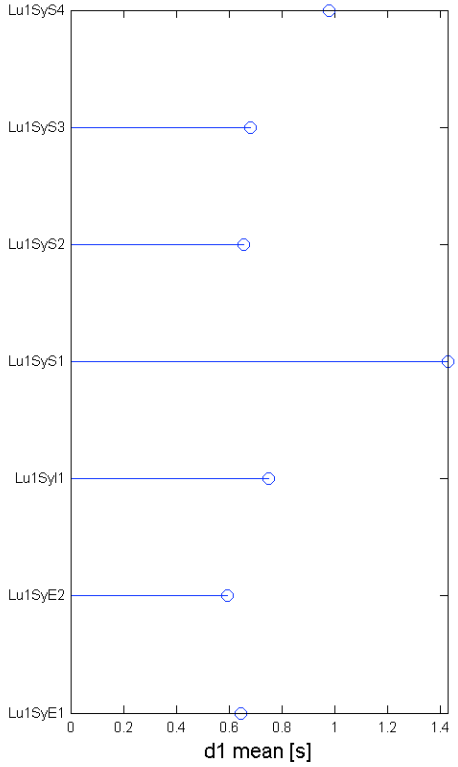


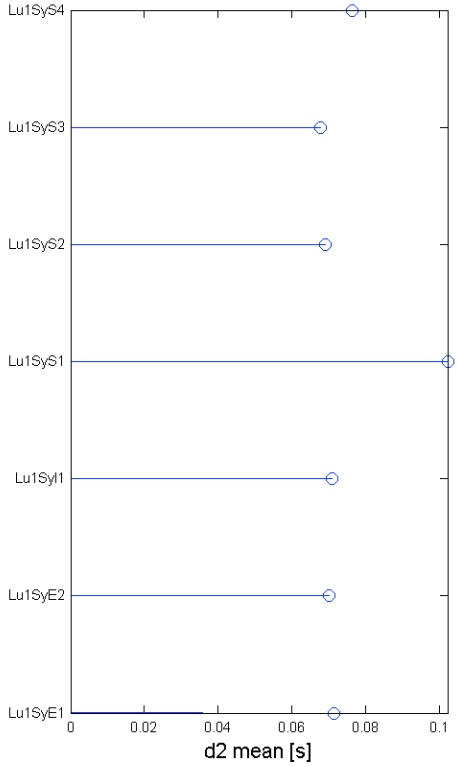
Pour chaque performance (pour Lu1, Mk1, MJ1 et MJ2), nous avons calculé d_1 et d_2 avec toute les autres performances des autres interprètes et les moyennes correspondantes. Ces moyennes figurent sur la figure ci-dessus. Note : pour le modèle temporel issu de la partition MIDI, d_1 et d_2 ont été calculées avec toutes les interprétations de tous les sujets.

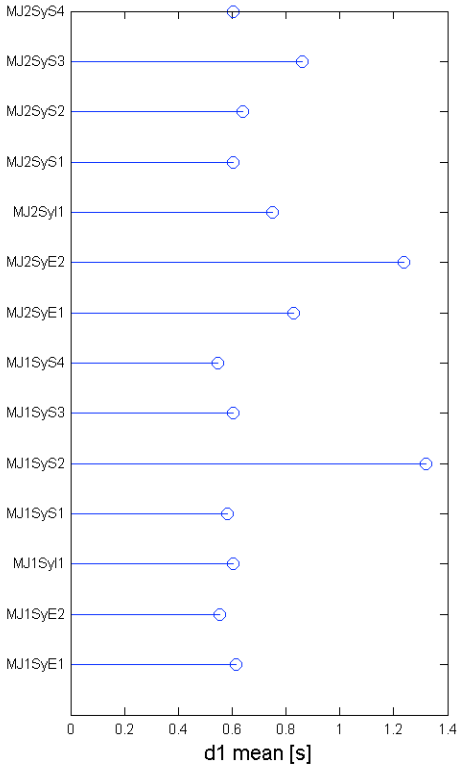
LuSyS2 peut servir de modèle temporel pour le time warping car elle minimise les 2 indicateurs de déformation infligée aux interprétations des autres interprètes.
Cependant nous utilisons aussi le modèle temporel provenant de la partition MIDI de la pièce, car il ne déforme pas beaucoup plus les interprétations et apparaît comme une interprétation plus 'neutre'.

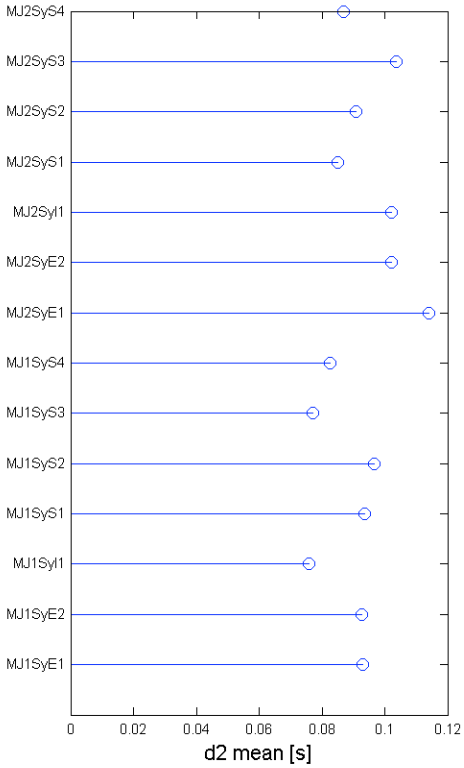
Avec LuSyS2 comme model, la déformation sur un segment est au pire de 1 s, ce qui reste acceptable. Mais la précision peut être améliorée lorsqu'on ne cherche pas à comparer plusieurs interprètes entre eux, mais plutôt plusieurs interprétations d'un même interprète. En effet des interprétations d'un même interprète sont souvent plus proches entre elles, dans leur séquencement temporel, qu'avec des interprétations d'autres interprètes.

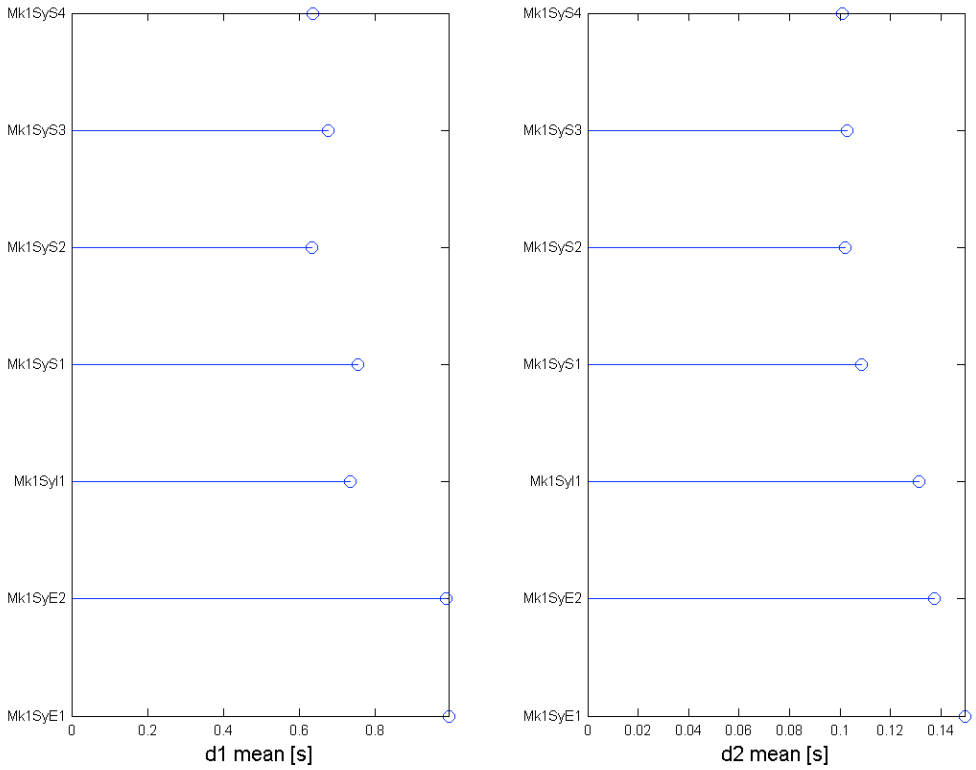
Aussi, lorsqu'on veut comparer des performances d'un même interprète, mieux vaut les conformer sur un model temporel propre à l'interprète. Nous avons donc répété le calcul de distance précédent pour chaque interprète, chaque calcul faisant intervenir uniquement ses propres performances. Les résultats apparaissent sur les figures ci-dessous.


Lu1SyS4
Lu1SyS3
Lu1SyS2
Lu1SyS1
Lu1SyI1
Lu1SyE2
Lu1SyE1
0 0.2 0.4 0.6 0.8 1 1.2 1.4
d1 mean [s]


Lu1SyS4
Lu1SyS3
Lu1SyS2
Lu1SyS1
Lu1SyI1
Lu1SyE2
Lu1SyE1
0 0.02 0.04 0.06 0.08 0.1
d2 mean [s]


MJ2SyS4
MJ2SyS3
MJ2SyS2
MJ2SyS1
MJ2SyI1
MJ2SyE2
MJ2SyE1
MJ1SyS4
MJ1SyS3
MJ1SyS2
MJ1SyS1
MJ1SyI1
MJ1SyE2
MJ1SyE1
0 0.2 0.4 0.6 0.8 1 1.2 1.4
d1 mean [s]


MJ2SyS4
MJ2SyS3
MJ2SyS2
MJ2SyS1
MJ2SyI1
MJ2SyE2
MJ2SyE1
MJ1SyS4
MJ1SyS3
MJ1SyS2
MJ1SyS1
MJ1SyI1
MJ1SyE2
MJ1SyE1
0 0.02 0.04 0.06 0.08 0.1 0.12
d2 mean [s]



Nous avons choisi l'interprétation qui minimise d_1 comme modèle pour chacun des interprètes.

Lu1SyE2 est l'interprétation qui sert de modèle temporel pour le time warping des performances de Lu1.
MJ1SyS4 est l'interprétation qui sert de modèle temporel pour le time warping des performances de MJ1 et MJ2.
Mk1SyS2 est l'interprétation qui sert de modèle temporel pour le time warping des performances de Mk1.

Nous avons sauvegardé les données traitées et time warpées sur ces modèles temporels dans des fichiers *.mat* pour ne pas avoir à refaire les calculs à chaque fois (cf paragraphe I-4 : *Sauvegarde des données*).

4) Fonctions Matlab

Les fonctions écrites en Matlab sont divisées en deux catégories : la première regroupe les fonctions de traitement des données (Preprocessing, Time Warping, …) la seconde comprend les fonctions de visualisation. Chaque catégorie est elle-même divisée en deux groupes : les fonctions de type générique, utilisables avec toutes sortes de données (par exemple une capture de mouvement avec un système Vicon…) et des fonctions plus spécifiques à l'Optotrak. Les fonctions plus spécifiques à l'Optotrak doivent cependant être 'assez facilement' adaptables pour d'autres systèmes d'acquisition de données de position comme le Vicon.

1) Fonctions de traitement des données

1) Génériques

Quelques fonctions génériques ont été crées pour traiter les données : dérivation et filtrage, time warping, et calcul d'angle.

1) **filterDeriv**

```
% TASKS PERFORMED:
%   - Takes a sampling rate and a data array and calculates the derivative
%   array : 1/fs*(x(n+1)-x(n)).
%   - Apply a low pass filter (y(n) = 0.8 y(n-1) + 0.2 x(n)) to smooth the
%   derivative (normalized cut-off frequency of this filter : fcn = 0.0344,
%   so the real cut-off frequency is fc = fcn*fs).
%   - Return an array containing the filtered derivative.
%
% SAMPLE USAGE:
%   filter_deriv_data = filterDeriv( fs, data )
```

2) **TimeWarp**

```
% TASKS PERFORMED:
%   - Takes in an array of data ('data'), the coresponding samplerate
%   ('fs'), the associated timing array in s ('timing') and a model
%   timming array ('timing_model') to conform the data.
%   - The given array of data is first truncated from the first to the
%   last timming values. Then it is time warped (i.e synchronized
%   to the given timing model) and resampled. The length of the output
%   array is : round(fs*timing_model(end)) - round (fs*timing_model(1)) + 1.
%   - Return an array of time warped and resampled data, and an array
%   containing the time values in s (after warping).
%   - This function uses the spline2 function which is derived from the
%   spline built-in function of MatLab and produce a optimised cubic
%   interpolation of the points.
%
```

```
% SAMPLE USAGE:
%    [warped_data, tw] = timeWarp( data, fs, timing, timing_model )
%
% WARNING:
%    Timing arrays ('timming' and 'timming_model') have to be both in s.
```

3) **CalculateAngle**

```
% TASKS PERFORMED:
%    Calculate the non-oriented (always positive) angle between the 2 given 3D
%    vectors u and v.
%
% SAMPLE USAGE:
%    angle = calculateAngle( u, v );
```

2) Optotrak

Nous décrivons ici 6 fonctions directement appelées par l'utilisateur et plus spécifiques à l'Optotrak. Elles réalisent tous les traitements relatifs aux fichiers .ndf correspondant à des performances données.

1) **CheckNDF**

```
% TASKS PERFORMED:
%    - Find erronous points (i.e values of less than -10000 or greater than
%    10000 due to occlusion of marker)in the given .ndf file.
%    - Return a 3-dimentional cell array containing on each line : the name
%    of the marker, the number of erronous points for this marker, and
%    a vector containing their indices in the data.
%
% SAMPLE USAGE:
%    errorC = checkNDF( 'MJ1SyS1.ndf' );
```

2) **checkSubjectPieceAllNDF**

```
% TASKS PERFORMED:
%    - apply the checkNDF function to the specified performances for
%    a given subject and a given piece of music.
%    - by default, the performances (each corresponding to a .ndf files)
%    are {'E1', 'E2', 'S1', 'S2', 'S3', 'S4', 'I1'}. The function return an
%    error if one of the ndf file can not be process.
%
% SAMPLE USAGE:
%    errorAllC = checkSubjectPieceAllNDF( 'Lu1Sy', {'E1','E2'} );
```

3) **processStoreNDF**

```
% TASKS PERFORMED:
%   - Takes the Optotrak data held in the given .NDF file, performs a
%   coordinate transform in order to correct for a skew in the data
%   produced by the Optotrak system in the McGill psychology labcleans it,
%   and stores the following information (as functions of time) in the
%   workspace:
%      - linear coordinate (3D + euclidian norm)
%      - velocity (3D + euclidian norm)
%      - acceleration (3D + euclidian norm)
%      - secousse (derivative of acceleration) (3D + euclidian norm)
%      - several angles, angular velocty and acceleration
%   - If a second parameter is given and equals to 1, then the data are
%   truncated from the begining of the first note to the begining of the
%   last note of the performance (usefull for synchronisation). To do
%   that, the timing file associated with the performance is automatically
%   loaded.
%
% SAMPLE USAGE:
%   processStoreNDF( 'MJ1SyS1.ndf', 1 );
%
% WARNING:
%   - when the second parameter is 1, the function will return an error if
%   it can't load the timing file associated to the performance you want
%   to process. So this timing file have to be in your Matlab path. It's
%   name has to be the same as the ndf file with 'Timing.txt' extantion.
%   For example if you call process('MJ1SyS1.ndf', 1) you need to have a
%   valid timing files in your MatLab path called 'MJ1SyS1Timing.txt'.
%   - Makes use of the saved tabledata.mat file in order to properly
%   correct the coordinates. This file must be in your MatLab path.
%   - WORK ONLY FOR MCGILL SUBJECTS : for first set of Strav performers
%   (W, R, ...) the fixCoordinateSkewC function is not necessary !!!!!!!!!!
```

4) **processStoreSubjectPieceAllNDF**

```
% TASKS PERFORMED:
%   - apply the processStoreNDF function to the specified performances for
%   a given subject and a given piece of music.
%   - by default, the performances (each corresponding to a .ndf files)
%   are {'E1', 'E2', 'S1', 'S2', 'S3', 'S4', 'I1'}. The function return an
%   error if one of the ndf file can not be process.
%
% SAMPLE USAGE:
%   processStoreSubjectPieceAllNDF( 'Lu1Sy', 1, {'E1','E2'} );
```

5) **processStoreTimeWarpNDF**

```
% TASKS PERFORMED:
%   - Does the same task as processStoreNDF except that it TIMEWARPS all
%   the data according to a given timing .txt file, before assigning them
%   to the workspace.
```

```
%
% SAMPLE USAGE:
%    processStoreTimeWarpNDF( 'MJ1SyS1.ndf', 'SyScoreTiming.txt' );
%
% WARNING:
%    - The function will return an error if it can't load the timing file
%    associated to the performance you want to process. So this timing file
%    have to be in your Matlab path. It's name has to be the same as the
%    ndf file with 'Timing.txt' extantion. For example if you try to
%    process 'MJ1SyS1.ndf', you need to have a valid timing files in your
%    MatLab path called 'MJ1SyS1Timing.txt'.
%    - Makes use of the saved tabledata.mat file in order to properly
%    correct the coordinates. This file must be in your MatLab path.
%    - WORK ONLY FOR MCGILL SUBJECTS : for first set of Strav performers
%    (W, R, ...) the fixCoordinateSkewC function is not necessary !!
```

6) **processStoreTimeWarpSubjectPieceAllNDF**

```
% TASKS PERFORMED:
%    - apply the processStoreTimeWarpNDF function to the specified
%    performances for a given subject and a given piece of music.
%    - by default, the performances (each corresponding to a .ndf files)
%    are {'E1', 'E2', 'S1', 'S2', 'S3', 'S4', 'I1'}. The function return an
%    error if one of the ndf file can not be process.
%    - if more than one performance is process for a same type (E, S or I),
%    the function will automatically calculate the mean and store it in the
%    workspace.
%
% SAMPLE USAGE:
%    processStoreTimeWarpSubjectPieceAllNDF( 'Lu1Sy', 'SyScoreTiming.txt', {'E1','E2'} );
%
% WARNING:
%    call the 'calculateMeanOptoPerf' function to calculate the mean of
%    several performances. If you change the field calculated by
%    'processStoreTimeWarp' (for example you add an angle...), you have to
%    report it in the 'calculateMeanOptoPerf' code !!
```

3) Fonctions de visualisation des données

1) Génériques

Trois fonctions génériques ont été crées pour représenter des données en fonction du temps, à la manière d'un oscilloscope. Ces 3 fonctions font la même chose, mais s'applique respectivement sur des données en 1, 2 ou 3 dimensions.

1) **movieDataCell**

```
% TASKS PERFORMED:
%    - Takes in a cell of array ('dataC') and creates a QuickTime movie
```

```
%    that shows the data as a function of time.
%    - The samplerate of the data ('fs') has to be specified.
%    - The number of frame per second of the movie ('fps') has to be
%    specified. Warning : 'fs' must be a multiple of 'fps' because there is
%    no resampling !
%    - At the end of the graphing run, a QuickTime file is saved with the
%    specified name ('movieName') in the MatLab current directory. The user
%    must wait until the end of the graphical display  for this file to be
%    properly created.
%    - A sound file ('soundName') synchronized to the  data has to be
%    specified to be added to the movie.
%    - The length of the time window displayed in the movie ('timeWinLength')
%    can be specified (in s). The default value is 4 s.
%    - An array of time can be specified in s to be vertically plot. Default
%    value is [].
%    - The legend of the movie can be set by legendC, default value is {}.
%    - 'yLabel' can be set, default value is ''. (xlabel is always t)
%    - the grid on/off can be set with the boolean parameter 'gridVal'.
%    - The 'dimension' of the movie can be specified in pixel. The axes limits
%    are automatically set to match the limits of the data.
%
% SAMPLE USAGE:
%    movieDataCell( dataC, fs, fps, movieName, soundName, timeWinLength, timing, legendC, yLabel,
gridVal, dimension )
%    movieDataCell( {sin(2*pi*1/200*[0:400]),cos(2*pi*1/200*[0:400])}, 100, 10, 'test.mov',
'nosound.wav', 2, [0 1 2 3 4], {'signal 1','signal 2'}, 'signal', 1, [560 420] )
```

2) **movieDataCell2D**

```
% TASKS PERFORMED:
%    - Does the same tasks as 'movieDataCell' function, except it gives a
%    2D parametric representation of the data. The first argument dataC is
%    a cell containing itself cells of 2 data arrays.
%    - 'timing' parameter is not available in movieDataCell2D.
%    - Axis labels can be set via 'axisLabelC'. Default value is {'x','y'}.
%    - Default 'timeWinLength' is 1s.
%
% SAMPLE USAGE:
%    movieDataCell2D( dataC, fs, fps, movieName, soundName, timeWinLength, legendC, axisLabelC,
gridVal, dimension )
%    movieDataCell2D(
{{sin(2*pi*1/200*[0:200]),cos(2*pi*1/200*[0:200])},{1/2*cos(2*pi*1/200*[0:200]),1/2*sin(2*pi*1/200*[
0:200])}}, 100, 10, 'test.mov', 'nosound.wav', 5, {'signal 1','signal 2'}, {'x','y'}, 1, [560 420] )
```

3) **movieDataCell3D**

```
% TASKS PERFORMED:
%    - Does the same tasks as 'movieDataCell' function, except it gives a
%    3D parametric representation of the data. The first argument dataC is
%    a cell containing itself cells of 3 data arrays.
%    - 'timing' parameter is not available in movieDataCell3D.
%    - Axis labels can be set via 'axisLabelC'. Default value is {'x','y','z'}.
```

```
%    - Default timeWinLength is 1s.
%    - Axes View can be set via the view parameter, default is [-37.5 30].
%
% SAMPLE USAGE:
%    movieDataCell3D( dataC, fs, fps, movieName, soundName, timeWinLength, legendC, axisLabelC,
gridVal, view, dimension )
%    movieDataCell3D(
{{sin(2*pi*1/200*[0:200]),sin(2*pi*1/200*[0:200]),cos(2*pi*1/200*[0:200])},{1/2*cos(2*pi*1/200*[0:20
0]),1/2*cos(2*pi*1/200*[0:200]),1/2*sin(2*pi*1/200*[0:200])}}, 100, 10, 'test.mov', 'nosound.wav', 5,
{'signal 1','signal 2'}, {'x','y','z'}, 1, [-30,30], [560 420] )
```

2) Optotrak

Nous décrivons aussi des fonctions pour la visualisation des données d'Optotrak (qui appellent les fonctions précédentes avec des paramètres spécifiques).

1) **movieOptoDataCell**

```
% TASKS PERFORMED:
%    - Takes in a cell of Optotrak data run names ('optoDataNameC') and
%    creates a QuickTime movie that shows the motion of the specified
%    markers. The Optotrak data run must be in the workspace (use
%    processStoreNDF or processStoreTimeWarpNDF for that).
%    - At the end of the graphing run, a QuickTime file is created with the
%    specified name ('movieName'). The user must wait until the end of the
%    graphical display  for this file to be properly created. The created
%    movie has 10 frame per second (but there is NO undersampling of data).
%    - A sound file ('soundName') synchronized to the Optotrak data has to
%    be specified to be added to the movie.
%    - The length of the time window displayed in the movie ('timeWinLength')
%    can be specified (in s). The default value is 4 s.
%    - An array of time in s ('timing_model'), typically corresponding to time
%    warping segments, can be specified to be vertically plot. Default
%    value is [].
%    - movieOptoDataCell enable to substract or not the mean of the data
%    before plotting them, via the boolean 'substractMean' parameter. Default
%    value is 1, that correspond to substract the mean.
%    - 'yLabel' can be set, default value is ''. (xlabel is always t)
%    - the grid on/off can be set with the boolean parameter 'gridVal'.
%    - The 'dimension' of the movie can be specified in pixel (default is
%    560*420). The axes limits are automatically set to match the limits of
%    the data.
%
% SAMPLE USAGE:
%    movieOptoDataCell(optoDataNameC, movieName, soundName, timeWinLength, timing_model,
substractMean, yLabel, gridVal, dimension)
%    movieOptoDataCell({'MJ1SyS1TW_5z_p', 'MJ1SyS1TW_1z_p'},'essai2.mov','son.wav',5,[],0,'z
[mm]',1,[600 500])
%
% 'movieOptoDataCell' uses the generic function 'movieDataCell'. See 'movieDataCell' for more infos
```

2) **movieOptoDataCell2D**

```
% TASKS PERFORMED:
%    - Does the same tasks as movieOptoDataCell function, except it gives a
%    parametric representation of the motion. The first argument
%    optoDataNameC is a cell containing itself cells of 2 Optotrak data run
%    names.
%    - 'timing_model' parameter is not available in movieOptoDataCell2D.
%    - Axis labels can be set via 'axisLabelC'. Default value is {'',''}.
%    - Default 'timeWinLength' is 1s.
%
% SAMPLE USAGE:
%    movieOptoDataCell2D(optoDataNameC, movieName, soundName, timeWinLength, substractMean,
axisLabelC, gridVal, dimension)
%
movieOptoDataCell2D({{'MJ1SyS1TW_5y_p','MJ1SyS1TW_5z_p'},{'MJ1SyS1TW_4y_p','MJ1SyS1TW
_4z_p'}},'essai2.mov','son.wav',2,0,{'y [mm]','z [mm]'},1,[600 500]);
%
%    see 'movieOptoDataCell' and 'movieDataCell2D' for more infos
```

3) **movieOptoDataCell3D**

```
% TASKS PERFORMED:
%    - Does the same tasks as movieOptoDataCell function, except it gives a
%    3D parametric representation of the motion. The first argument
%    optoDataNameC is a cell containing itself cells of 3 Optotrak data run
%    names.
%    - 'timing_model' parameter is not available in movieOptoDataCell3D.
%    - Axis labels can be set via 'axisLabelC'. Default value is {'x','y','z'}.
%    - Axes View can be set via the 'view' parameter, default is [-37.5 30].
%    - Default 'timeWinLength' is 1s.
%
% SAMPLE USAGE:
%    movieOptoDataCell3D(optoDataNameC, movieName, soundName, timeWinLength, substractMean,
axisLabelC, gridVal, view, dimension)
%    movieOptoDataCell3D({{'MJ1SyS1TW_5x_p','MJ1SyS1TW_5y_p','MJ1SyS1TW_5z_p'},
{'MJ1SyS1TW_4x_p','MJ1SyS1TW_4y_p','MJ1SyS1TW_4z_p'}},'essai3.mov','son.wav',2,0,{'x[mm]','y[m
m]','z[mm]'},1,[-30 30],[400 600]);
%
%    see 'movieOptoDataCell' and 'movieDataCell3D' for more infos
```

4) **plotOptoDataCell**

```
% TASKS PERFORMED:
%    - Takes in a cell of an Optotrak data run name ('optoDataNameC') and
%    plot the motion of the specified markers. The Optotrak data run must
%    be in the workspace (use 'processStoreNDF' or 'processStoreTimeWarpNDF'
%    for that).
%    - An array of time in s ('timing_model'), typically corresponding to time
%    warping segments, can be specified to be vertically plot. Default
%    value is [].
%    - 'plotOptoDataCell' enable to substract or not the mean of the data
```

```
%    before plotting them, via the boolean 'subtractMean' parameter. Default
%    value is 1, that correspond to substract the mean.
%    - The title of the plot is set by the 'plotTitle' parameter
%    - 'yLabel' can be set, default value is ''. (xlabel is always t)
%    - The boolean 'subPlot' parameter enable to do subplot.
%    - The boolean 'color' parameter enable to plot in color or black&white
%    - The boolean 'gridVal' parameter enable to draw the grid
%    - X-coordinate limites 'xlimit' can be set with a vector of 2 values
%    in [s]. Default value is set to plot all the motion.
%
% SAMPLE USAGE:
%    plotOptoDataCell( optoDataNameC, timing_model, subtractMean, plotTitle, yLabel, subPlot, color,
gridVal, 'xlimit')
%    plotOptoDataCell({'MJ1SyS1_5z_p', 'MJ1SyS1_1z_p'},[100 500 1000],1,'MJ1 2 markers','position z
[mm]',1,1,1)
```

5) **plotOptoDataCell2D**

```
% TASKS PERFORMED:
%    - Does the same tasks as plotOptoDataCell function, except it gives a
%    2D parametric representation of the motion. The first argument
%    optoDataNameC is a cell containing itself cells of 2 Optotrak data run
%    names.
%    - timing_model parameter is not available in plotOptoDataCell2D.
%    - Axis labels can be set via plotAxisLabelC. Default value is {'','}.
%
% SAMPLE USAGE:
%    plotOptoDataCell2D( optoDataNameC, subtractMean, plotTitle, plotAxisLabelC, subPlot, color,
gridVal )
%
plotOptoDataCell2D({{'MJ1SyS1TW_5y_p','MJ1SyS1TW_5z_p'},{'MJ1SyS1TW_4y_p','MJ1SyS1TW_4
z_p'}},0,'essai',{'y [mm]','z [mm]'},0,1,1)
%
%    see plotOptoDataCell for more infos
```

6) **plotOptoDataCell3D**

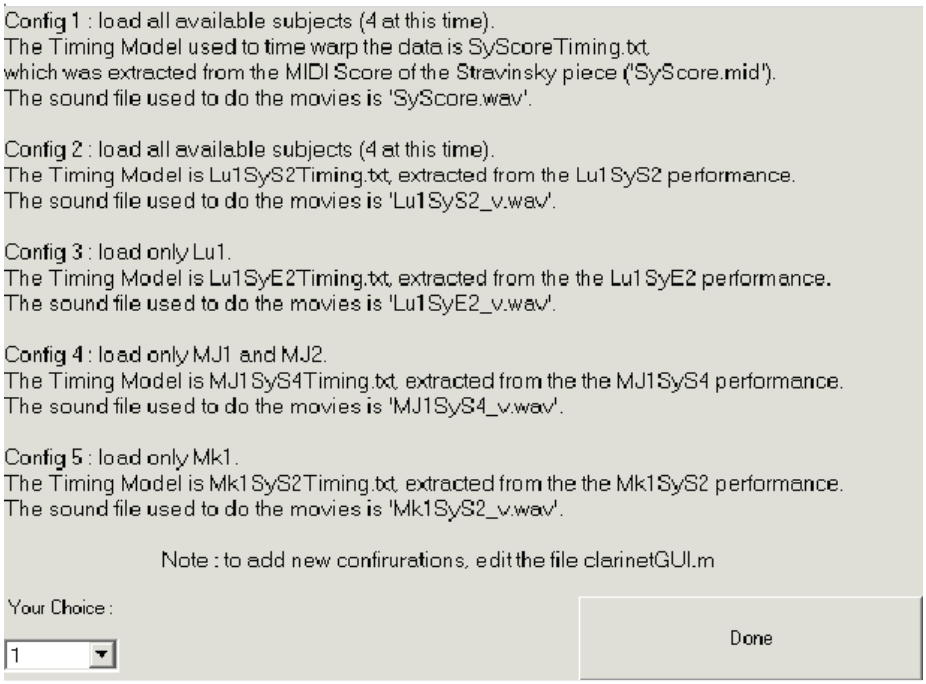
```
% TASKS PERFORMED:
%    - Does the same tasks as plotOptoDataCell function, except it gives a
%    3D parametric representation of the motion. The first argument
%    optoDataNameC is a cell containing itself cells of 3 Optotrak data run
%    names.
%    - timing_model parameter is not available in plotOptoDataCell3D.
%    - Axis labels can be set via plotAxisLabelC. Default value is {'x','y','z'}.
%
% SAMPLE USAGE:
%    plotOptoDataCell3D( optoDataNameC, subtractMean, plotTitle, plotAxisLabelC, subPlot, color,
gridVal )
%
plotOptoDataCell3D({{'MJ1SyS1TW_5x_p','MJ1SyS1TW_5y_p','MJ1SyS1TW_5z_p'},{'MJ1SyS1TW_4
x_p','MJ1SyS1TW_4y_p','MJ1SyS1TW_4z_p'}},0,'essai',{'x [mm]','y [mm]','z [mm]'},0,1,1)
```

5) **clarinetGUI**

Une interface graphique à été crée pour pouvoir utiliser simplement toutes ces fonctions de visualisation sur les données de clarinette de McGill.

Plusieurs configurations sont possibles pour la GUI : on peut charger un nombre variable de sujets (4 sont disponibles pour l'instant), préalablement warpés sur différents modèles temporels et sauvés dans des fichiers *.mat*. Lors du lancement de la GUI (par la commande 'clarinetGUI' dans la console MatLab) une première fenêtre apparaît pour permettre de choisir entre 5 configurations. On choisit une des configurations et les fichiers *.mat* sont automatiquement chargés, sous réserve qu'ils existent et qu'ils soient dans le 'path' de MatLab. Les noms des fichiers chargés apparaissent dans la console (ainsi que d'éventuels messages d'erreur si des fichiers ne sont pas trouvés).

Voici la fenêtre qui apparaît pour le choix de la configuration :



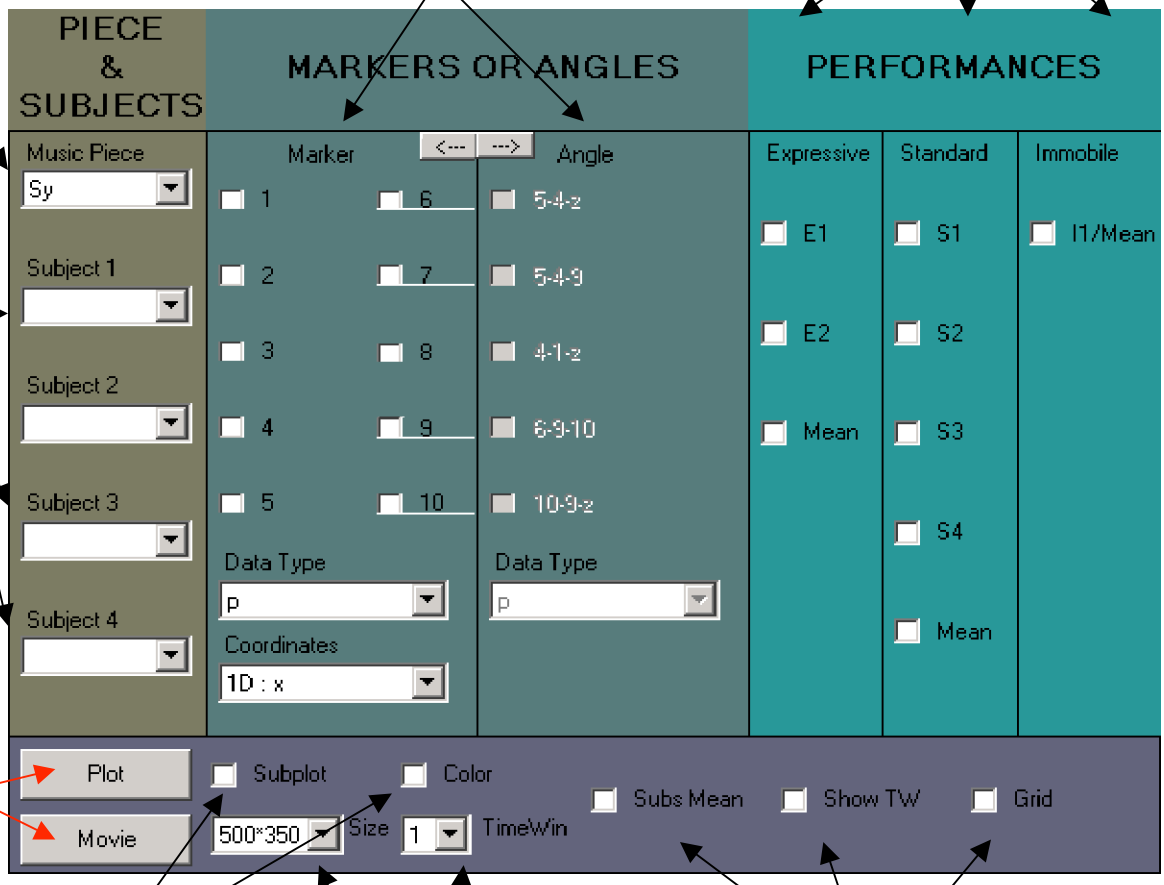
Il est possible de créer d'autres fichiers .mat (avec d'autres modèles pour le time warping par exemple) et d'y associer d'autres configurations pour la GUI. Il faut pour cela éditer à la main le code 'clarinet.m'. La partie de code à modifier est indiquée au début du fichier.
Il est aussi possible d'afficher de nouvelles données (autre angle, nouveau calcul de norme,…). Pour cela il faut rentrer un peu plus dans le fichier 'clarinet.m'. Des indications sont données à la ligne 126 du code.

Et voici la GUI proprement dite :

Choix des données à afficher pour chaque sujet. Possibilité d'afficher plusieurs markers OU plusieurs angles en même temps.
Note : les 2 flèches en gris permettent de passer du « mode marker » au « mode angle ».
Pour les « Data Type », 'p' correspond à la position, 'v' à la vitesse, 'a' à l'accélération et 's' à la secousse (dérivée de l'accélération)

Checkboxes pour sélectionner les interprétations à afficher

oix du morceau de sique (actuellement seul avinsky est disponible)



ossibilité 'afficher 4 ujets en même emps

sibilité faire un ot' ou un ovie'

tions pour le 'plot' :
- Subplot : affiche chaque donnée dans un graphe différent
- Color : affiche en couleur ou en noir et blanc

Options pour le 'movie' :
- Size : permet de choisir les dimensions du film créé
- TimeWin : permet de choisir la durée de la fenêtre temporelle représentée à chaque instant dans le film

Options pour le 'plot' ET le 'movie' :
- Subs Mean : soustrait la valeur moyenne des données avant de les afficher
- Show TW : affiche des lignes verticales aux instants c time warping
- Affiche la grille sur les graphes

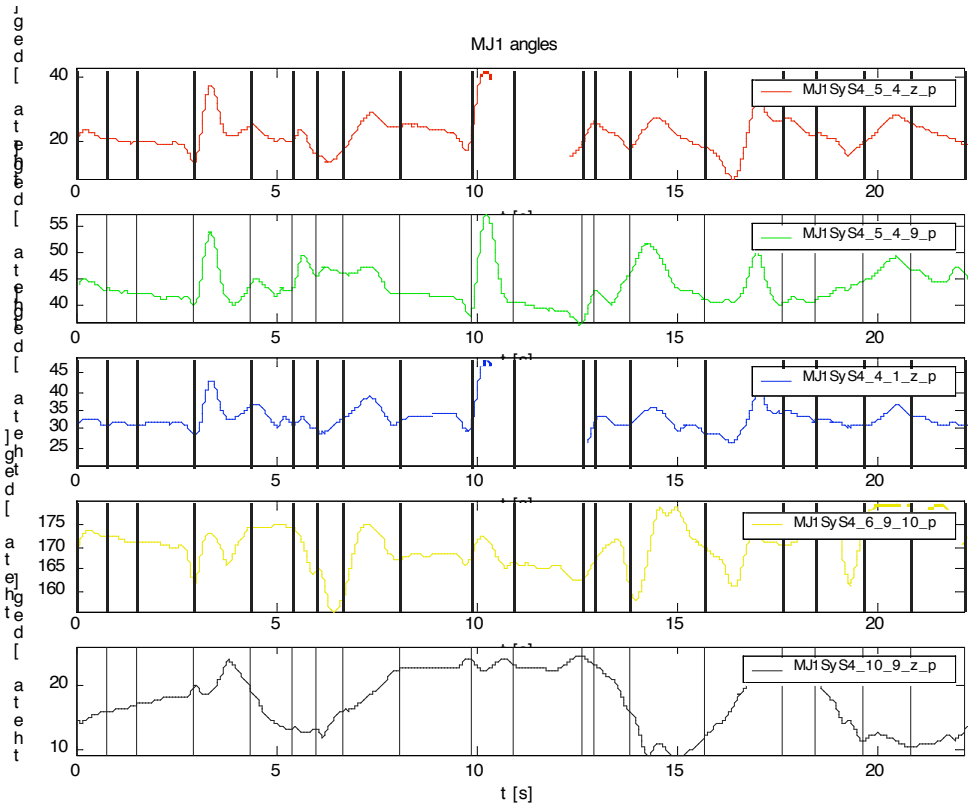
A faire : Amélioration des fonctions plotOptoDataCell

Les fonctions de visualisation plotOptoDataCell, plotOptoDataCell2D et plotOptoDataCell3D peuvent être améliorées.
Elles ne permettent pas de choisir la taille de la fenêtre temporelle pour laquelle les données sont représentées. Cela serait pourtant bien pratique pour pouvoir se concentrer sur une partie précise du mouvement. De plus elles ne sont pas génériques (fréquence d'échantillonnage fixée à 100 Hz pour l'Optotrak,…) et ne peuvent donc pas être utilisées avec d'autres système comme le Vicon.

➔ Il faudrait créer des fonctions génériques plotDataCell, plotDataCell2D et plotDataCell3D en s'inspirant des fonctions movieDataCell, movieDataCell2D et movieDataCell3D qui elles possèdent le paramètre de taille de fenêtre ('timeWinLength') et sont génériques.
Il faudrait ensuite modifier le code de plotOptoDataCell, plotOptoDataCell2D et plotOptoDataCell3D pour qu'elles appellent simplement plotDataCell, plotDataCell2D et plotDataCell3D avec des paramètres fixés pour l'Optotrak (tout comme movieOptoDataCell appelle movieDataCell).

5) Exemples d'utilisation des fonctions Matlab

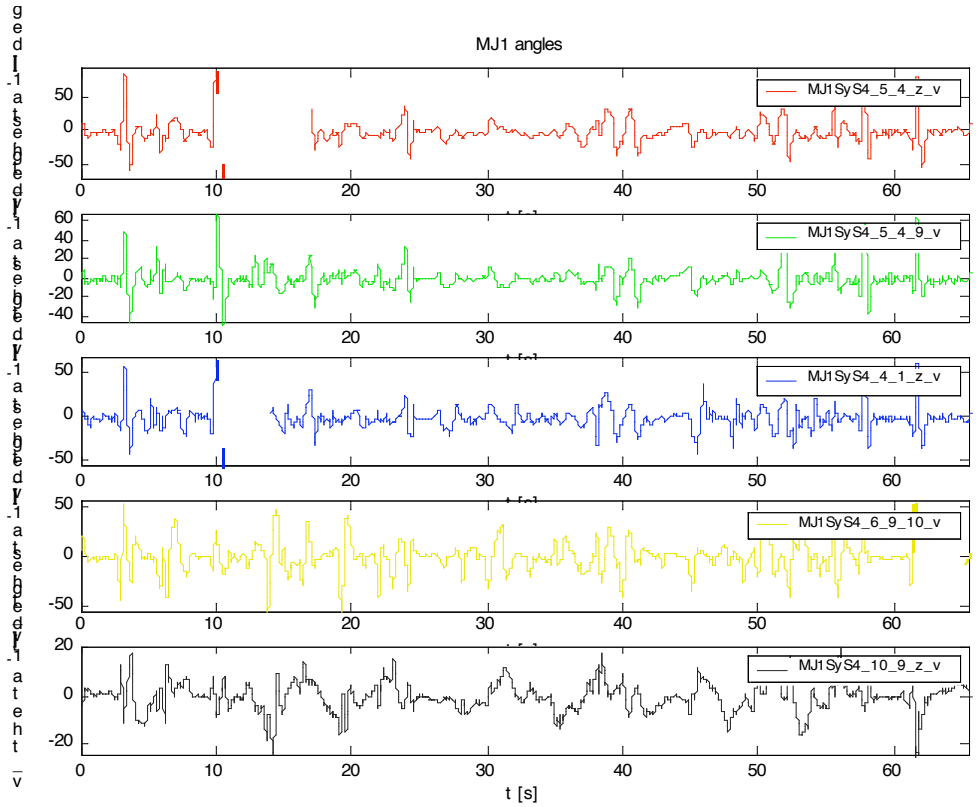
Nous donnons ici quelques figures réalisées à partir des données de clarinette de McGill et des fonctions MatLab de traitement et de visualisation décrites dans le paragraphe précédent.



Angles pour l'interprétation Standard 4 de MJ1
(fonction 'plotOptoDataCell')

% Code Matlab

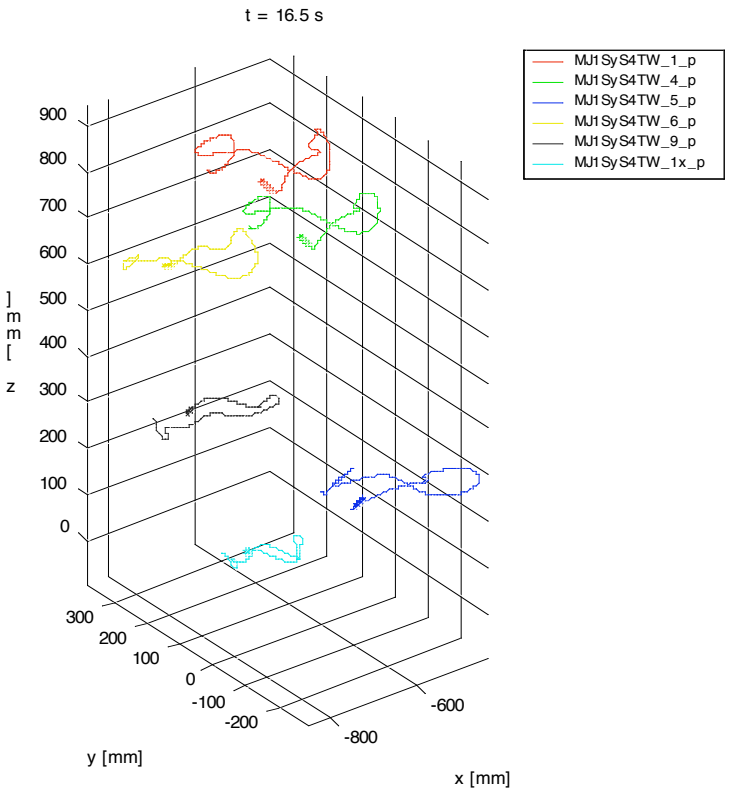
```
processStoreNDF('MJ1SyS4.ndf',1);
timing = load('MJ1SyS4Timing.txt');
timing = timing - timing(1); % force the timing to begin at 0
plotOptoDataCell({'MJ1SyS4_5_4_z_p', 'MJ1SyS4_5_4_9_p', 'MJ1SyS4_4_1_z_p', 'MJ1SyS4_6_9_10_p',
'MJ1SyS4_10_9_z_p'}, timing, 0, 'MJ1 angles','theta [deg]',1,1,0,[timing(1) timing(20)])
```



Vitesses angulaires pour la performance Standard 4 de MJ1
(fonction 'plotOptoDataCell')

% Code Matlab

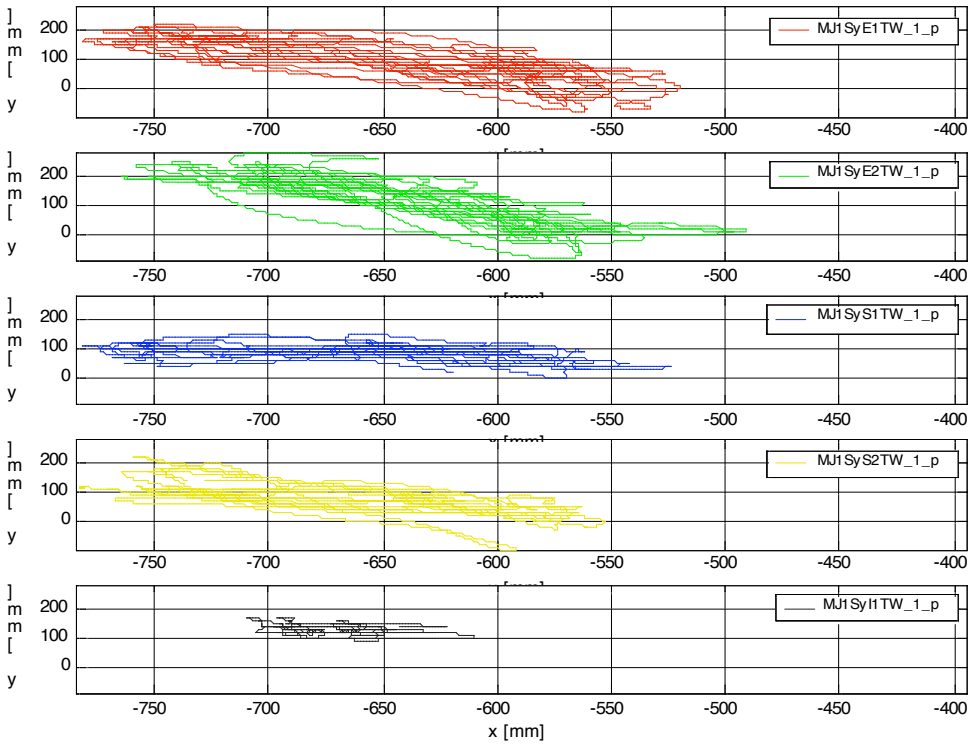
processStoreTimeWarpNDF('MJ1SyS4.ndf','SyScoreTiming.txt');
plotOptoDataCell({'MJ1SyS4_5_4_z_v', 'MJ1SyS4_5_4_9_v', 'MJ1SyS4_4_1_z_v', 'MJ1SyS4_6_9_10_v', 'MJ1SyS4_10_9_z_v'}, [], 0, 'MJ1 angles','v_theta [deg.s^-1]',1,1,0)



Représentation en 3D des 6 markers pour MJ1SyS4
(fonction 'movieOptoDataCell3D')

% Code Matlab

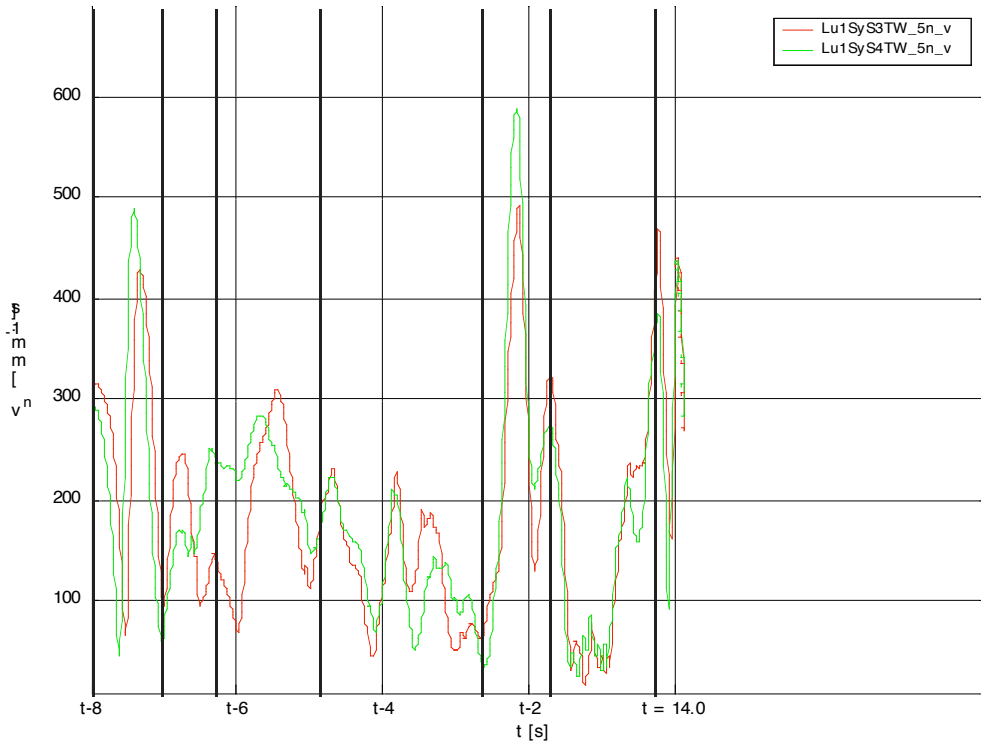
```
load('MJ1Sy_TW_MJ1SyS4.mat');
movieOptoDataCell3D({{'MJ1SyS1TW_1x_p','MJ1SyS1TW_1y_p','MJ1SyS1TW_1z_p'},...
      {'MJ1SyS1TW_4x_p','MJ1SyS1TW_4y_p','MJ1SyS1TW_4z_p'},...
      {'MJ1SyS1TW_5x_p','MJ1SyS1TW_5y_p','MJ1SyS1TW_5z_p'},...
      {'MJ1SyS1TW_6x_p','MJ1SyS1TW_6y_p','MJ1SyS1TW_6z_p'},...
      {'MJ1SyS1TW_9x_p','MJ1SyS1TW_9y_p','MJ1SyS1TW_9z_p'},...
      {'MJ1SyS1TW_10x_p','MJ1SyS1TW_10y_p','MJ1SyS1TW_10z_p'}},...
  'demo.mov','MJ1SyS4_v.wav',4,0,{'x[mm]','y[mm]','z[mm]'},1,[-37.5 30],[600 600]);
```



Représentation du mouvement dans le plan (Oxy) du marker 1
pour plusieurs performances de MJ1 (E1, E2, S1, S2, I1)
(fonction 'plotOptoDataCell2D')

% Code Matlab

```
load('MJ1Sy_TW_MJ1SyS4.mat');
plotOptoDataCell2D({{'MJ1SyE1TW_1x_p','MJ1SyE1TW_1y_p'},...
      {'MJ1SyE2TW_1x_p','MJ1SyE2TW_1y_p'},...
      {'MJ1SyS1TW_1x_p','MJ1SyS1TW_1y_p'},...
      {'MJ1SyS2TW_1x_p','MJ1SyS2TW_1y_p'},...
      {'MJ1SyI1TW_1x_p','MJ1SyI1TW_1y_p'}}, 0,'essai',{'x [mm]','y [mm]'},1,1,1)
```



Norme de la vitesse du marker 5 en fonction du temps
pour 2 interprétations de Lu1 (S3 et S4)
(fonction 'movieOptoDataCell')

Code Matlab

```
load('Lu1Sy_TW_SyScore.mat');
timing = load('SyScoreTiming.txt');
timing = timing - timing(1); % force the timing to begin at 0
movieOptoDataCell({'Lu1SyS3TW_5n_v', 'Lu1SyS4TW_5n_v'},'demo.mov','SyScore.wav',6,timing,0,'v_n
[mm.s^-1]',1,[600 450])
```

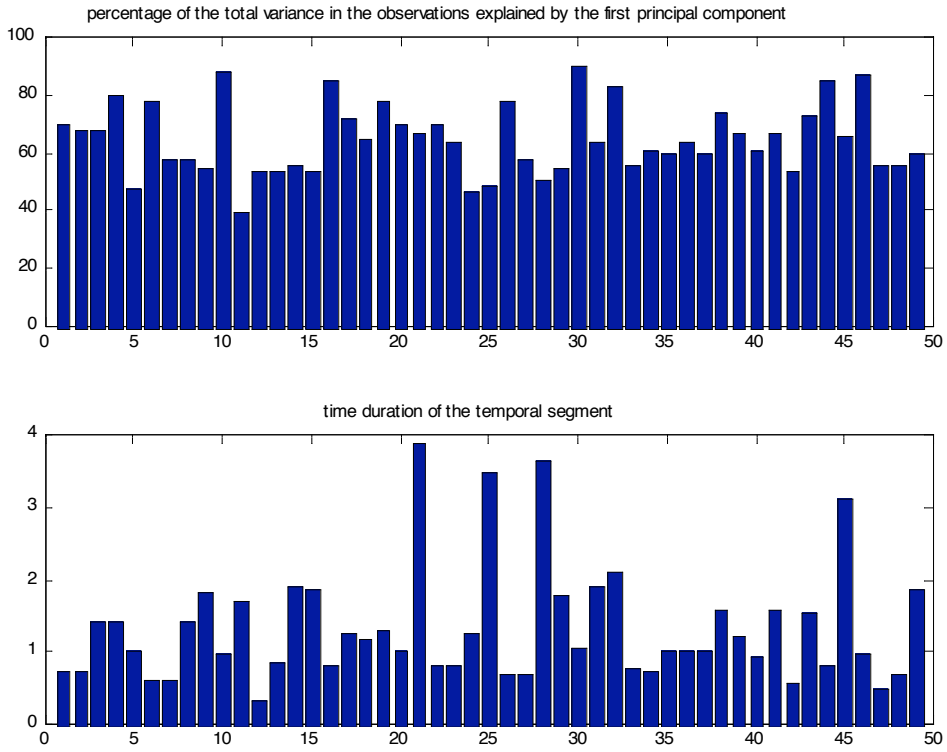
6) Perspectives

Les travaux effectués ont permis la réalisation d'outil pour le traitement et la visualisation de données gestuelles. Ces outils ne font pas eux-mêmes d'analyse, mais aide leur utilisateur à mieux visualiser le mouvement, pour mieux l'analyser par la suite.
La prochaine étape est donc de développer des techniques d'analyse du geste, d'extraction de paramètres pertinents du mouvement.

1) Analyse en composantes principales

Nous donnons ici une piste qui a été brièvement explorée, mais qui mériterait sûrement d'être approfondie par la suite : utiliser l'Analyse en Composantes Principales (ACP) sur les données de différents markers d'Optotrak pour évaluer la 'complexité' d'un geste.

Reprenons la segmentation temporelle de la partition de Stravinsky utilisée pour le time warping (cf p.11). Nous avons fait une analyse en composantes principales sur la norme de la vitesse des 6 markers de MJ1SyS1, pour chaque segment temporel. On peut penser que plus le mouvement est 'complexe', plus les normes des vitesses vont varier différemment les unes des autres dans le temps. Ainsi la variance expliquée par la première composante de l'ACP pourrait servir à évaluer la 'complexité' du geste. La figure ci-dessous donne en haut :la variance expliquée par la première composante de l'ACP, pour les 49 segments temporelles et en bas : la longueurs des segments en s.



Plus les segments sont courts, plus il y a de chance pour que la variance expliquée soit élevée. Cependant, la figue montre que cette relation n'est pas toujours vérifiée. Il pourrait être intéressant de comparer ces résultats avec une évaluation perceptive de la 'complexité' du mouvement à partir des vidéos.

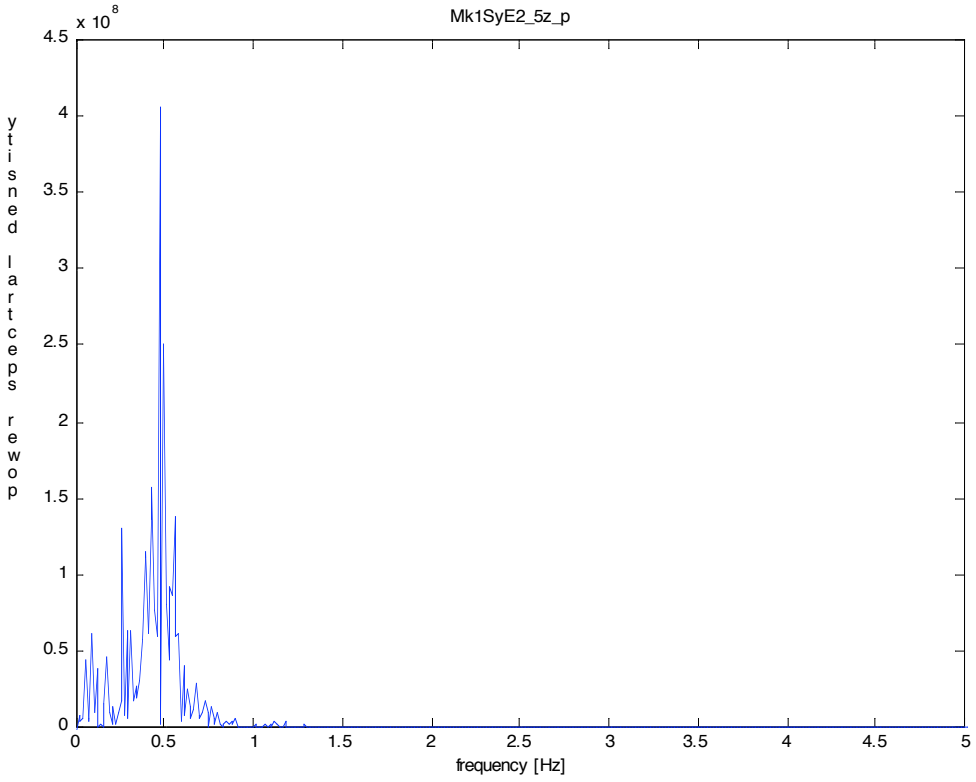
L'utilisation de l'ACP est 'à creuser'. Cette technique pourrait être utile pour des taches de segmentation automatique de geste, de reconnaissance de geste, et d'extraction de paramètres du geste (complexité,…).

2) Etude dans le domaine spectral

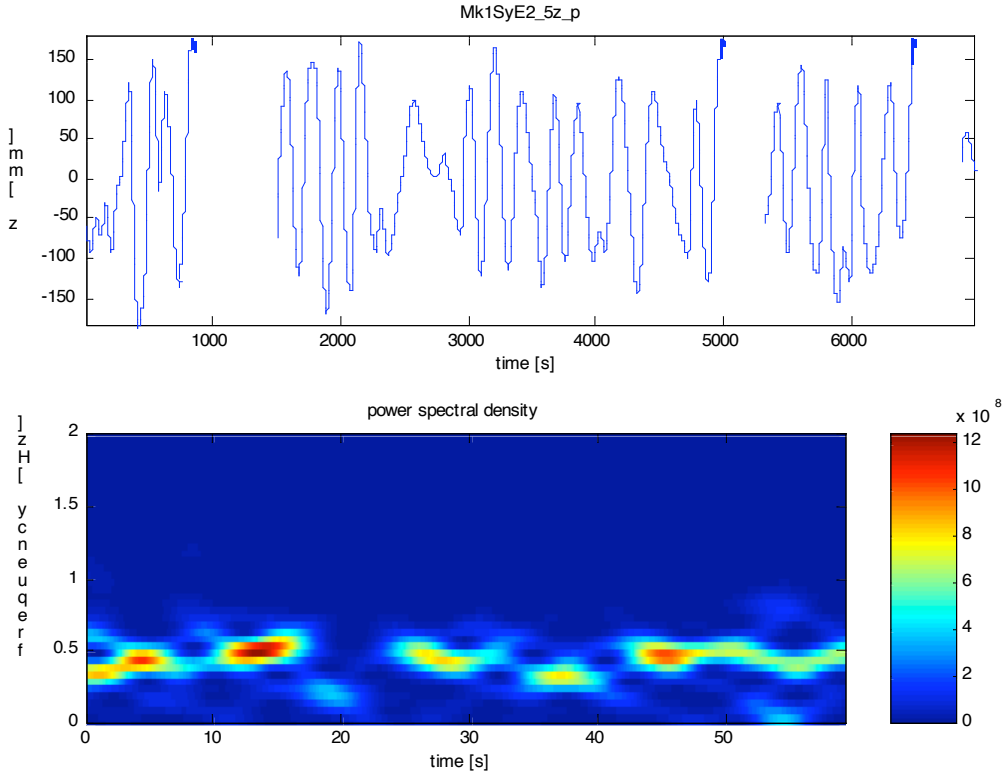
Nous avons également trouvé des propriétés intéressantes en examinant les données gestuelles de clarinette dans le domaine spectral.

Sur le marker 5z notamment (mouvement vertical du pavillon de la clarinette) nous obtenons pour certains joueurs (surtout Mk1) une très forte composante spectrale vers 0,5 Hz.

La figure ci-dessous donne la densité spectrale d'énergie calculée sur le marker 5z, sur toute l'interprétation Expressive 2 du sujet Mk1.



La fréquence de la composante spectrale dominante évolue au long du morceau, comme on le voit sur le spectrogramme ci-dessous. La fenêtre utilisée pour le spectrogramme est une fenêtre de Hann de 10 s.



7) Conclusion

Ce travail a permis d'établir des outils de traitement et de visualisation de données gestuelles acquises par Optotrak. La bibliothèque de fonctions Matlab est principalement dédiée aux mesures de gestes de clarinettistes faites à McGill, mais plusieurs fonctions génériques peuvent être utilisées sur des données gestuelles diverses (autres instruments, danse,…) capturées par Optotrak, Vicon ou autre…

Les outils créés constituent une première étape pour l'aide à l'analyse du mouvement. Des exemples d'utilisation sont donnés dans ce rapport, et quelques pistes sont posées pour l'analyse du geste.

Pour finir, merci beaucoup à Marcelo pour l'accueil tellement chaleureux, et à toute l'équipe de MusicTech !