

---

**REDS**  
*Static Hand Gestures*

---

Céline Hanouti  
Léane Salais

Master 2 DAC  
2020-2021

# Introduction

L'objectif du projet ce semestre est d'analyser le dataset *UC2017, Static and Dynamic Hand Gestures*. Notre travail se fonde sur une étude qui contextualise son usage. Dans cette étude, un panel de volontaires utilise sa main gauche pour signifier des ordres à un robot, et on enregistre la position des mains ;

**L'idée est d'entraîner le robot de sorte qu'il puisse généraliser et comprendre n'importe quel utilisateur à l'avenir.**

Les données de la base sont issues de deux sources jumelées - un tracker magnétique et un gant intelligent. Grâce à ce tracker et aux capteurs du gant, l'équipe de recherche peut prendre des instantanés d'un geste ou des captations à long terme sur un mouvement. On accède ainsi à de l'information concernant la position absolue d'une main dans l'espace (tracker) et le statut de ses vingt-deux articulations (gant).



Figure 1 – Plateforme expérimentale.  
Le volontaire porte le gant et le Tracker est visible sur son support tout à gauche.

Même si l'article analyse également les *mouvements* effectués par les volontaires, nous nous concentrons ici sur les *gestes statiques*, nommément sur la sous-base SG. L'information conservée pour chaque prise de vue contient l'identifiant du cobaye, les descripteurs de son geste selon les deux capteurs, et l'étiquette attribuée que le robot doit savoir reconnaître en toutes circonstances.

**Il s'agira de développer une classification des gestes de SG via des algorithmes d'apprentissage supervisé. L'étude étant orientée vers la robotique sociale, le modèle choisi devra remplir des critères de vitesse, de simplicité et de légèreté : la tâche de scan et de reconnaissance doit être instantanée. On vise également un score quasi-parfait, car le robot ne doit jamais présenter de comportements inexplicables.**

Ce rapport résume nos essais pour cette tâche.

En apportant un regard humain, en essayant de bien interpréter les enregistrements, nous pensons que nous pouvons obtenir un résultat similaire sinon meilleur que celui de l'équipe. Il faudra composer avec l'information assez incomplète de l'article - d'où les analyses approfondies. Quant à la mise en oeuvre, nous irons toujours aux modèles les plus simples ; nous sommes convaincues que pour une si petite base, l'utilisation de modèles coûteux serait contre-productive.

# Table des matières

<b>Introduction</b>	<b>1</b>
<b>1 Exploration. Analyser les données</b>	<b>3</b>
1.1 Perspective globale . . . . .	3
1.2 Interprétation des descripteurs . . . . .	4
1.2.1 Capteurs du gant . . . . .	4
1.2.2 Positions du Tracker . . . . .	4
1.3 Visualisation rapide des classes . . . . .	5
1.4 Conclusion : résumé sur la base . . . . .	6
<b>2 Définition. Ecrire un protocole</b>	<b>6</b>
2.1 Choix du mode d'évaluation . . . . .	6
2.2 Choix du classifieur utilisé . . . . .	7
2.3 Choix des dimensions d'entrée . . . . .	7
2.3.1 Suppression de <i>features</i> : filtrage . . . . .	8
2.3.2 Combinaison de <i>features</i> : PCA . . . . .	8
2.4 Conclusion : proposition de protocole . . . . .	10
<b>3 Contextualisation. Etablir des baselines</b>	<b>10</b>
3.1 Modèles simplistes . . . . .	11
3.1.1 Classification aléatoire . . . . .	11
3.1.2 Classification monodimensionnelle . . . . .	11
3.1.3 Classification en pleine dimensionalité . . . . .	12
3.2 Améliorations des baselines . . . . .	12
3.2.1 Hyperparameters tuning . . . . .	12
3.2.2 Méthodes d'ensemble . . . . .	13
3.3 Remarque méthodologique . . . . .	14
3.4 Conclusion : discussion des objectifs . . . . .	15
<b>4 Application. Mise en oeuvre du protocole</b>	<b>16</b>
4.1 Tâche principale . . . . .	16
4.1.1 Etude approfondie de la <i>feature selection</i> . . . . .	16
4.1.2 GridSearch et détection des paramètres optimaux . . . . .	18
4.1.3 Etude du modèle optimal . . . . .	18
4.2 Extension : classification monoutilisateur . . . . .	20
4.3 Extension : suppression du Tracker . . . . .	20
<b>Le mot de la fin</b>	<b>22</b>

# 1 Exploration. Analyser les données

Notre première tâche est de comprendre le format et le contenu de la base. De type .h5, elle compte 2400 enregistrements. Pour chacun, 29 descripteurs *Predictors* sont associés à deux étiquettes *Target* et *User*; on connaît donc le type de geste, unique, réalisé par un utilisateur unique.

Combien de gestes peut-on reconnaître dans la base, combien d'utilisateurs? Comment justifier le grand nombre de *Predictors* et à quoi correspondent-ils physiquement? Pour le savoir, nous recouperons les résultats de nos analyses et les informations (incomplètes) de l'article.

## 1.1 Perspective globale

L'article indique que les types de gestes sont associés à des nombres entiers de 1 à 24. Les enregistrements sont répartis uniformément par batches de 100 dans les 24 catégories. Il donne aussi la traduction de ces nombres en gestes, qui y est représentée par la figure ci-dessous.



Figure 2 – Visualisation des étiquettes *Target* des gestes

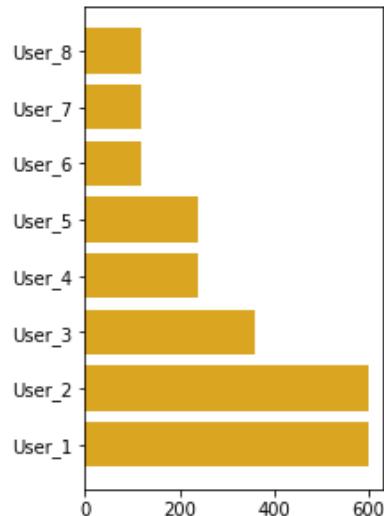


Figure 3 – Répartition des utilisateurs *User* sur la base

Par contre, la répartition des utilisateurs est déséquilibrée et illustrée par l'histogramme ci-dessus. Ils sont huit au total, dont deux majoritaires : avec 600 enregistrements chacun, les *Users* 1 et 2 représentent 50% de la base.

Nom	Type	Intervalle	Répartition
<i>Target</i>	Entier	{1 : 24}	Uniforme par batches de 100 gestes
<i>Users</i>	Entier	{1 : 8}	Déséquilibrée (plus ou moins de tests selon les cobayes)

Tableau 1 – Récapitulatif de la distribution des métadonnées

Sur les 29 colonnes des *Predictors*, les descripteurs des gestes, l'article reste cependant peu clair. Nous avons donc récupéré le type de données, l'intervalle décrit et le compte des valeurs prises pour chacune pour essayer de les interpréter.

Toutes les données sont numériques, mais l'éventail des possibilités n'est pas comparable partout. Il est plus large sur les 7 premières colonnes et plus restreint sur les 22 autres. Nous avons intuité que cette restriction venait d'une discréétisation, et après plusieurs étapes de vérification, établi que c'était le cas : il s'agit de décimaux et d'entiers. Cette séparation sert de base à notre interprétation.

Colonne	Type	Intervalle	Décompte	Discréétisation
1 - 7	Numérique	très divers	1500 à 2500 valeurs	Décimaux
8 - 29	Numérique	autour de [0, 250]	100 à 200 valeurs	Entiers

Tableau 2 – Typologie des descripteurs

## 1.2 Interprétation des descripteurs

### 1.2.1 Capteurs du gant

Le gant intelligent utilisé (CyberGlove II) comporte 22 capteurs d'angle. Ils s'activent pendant la prise de vue. Or, les entiers des colonnes *Predictors* 8 à 29 sont tous positifs et très largement bornés par 360 (la rotation totale étant humainement impossible). Ces 22 colonnes semblent bien correspondre aux angles d'articulations relevés par les capteurs, dont la carte exacte est donnée ci-dessous.

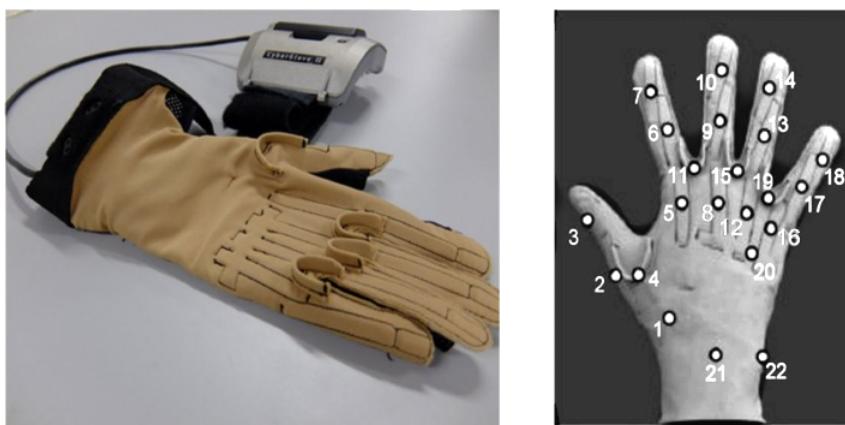


Figure 4 – Localisation des capteurs du CyberGlove II

### 1.2.2 Positions du Tracker

Par élimination, les décimaux doivent être les données recueillies par le Tracker. Il est censé renvoyer une position en 3D, on s'attend donc à avoir six colonnes - une position absolue et une orientation sur des axes X,Y et Z, soit deux fois trois.

Là, l'existence d'une septième colonne est inexpiquée. La documentation de la base et le [manuel](#) du Tracker ne sont pas informatifs. Nous soupçonnons la colonne n°4 d'être l'intruse : elle présente une normalisation différente de toutes les autres.

Colonne	Intervalle de normalisation	Interprétation incertaine
1, 2 et 3	aucune normalisation	Positions absolues sur X, Y, Z
4 seule	entre 0 et 1	Rôle inconnu
5, 6 et 7	entre -1 et 1	Rotations relatives sur X, Y, Z

Tableau 3 – Distribution des normalisations sur les colonnes Tracker

### 1.3 Visualisation rapide des classes

Nous sommes allées le plus loin possible dans l'interprétation, nous avons désor mais une image globale. Reste à visualiser les gestes dans l'espace des *Predictors*.

Pour donner une idée du degré de confusion entre eux et ramener l'espace 29-dimensionnel à quelque chose de lisible, nous effectuons une PCA à trois caractères. L'essai sert juste à détecter des groupes de classes qui se positionneraient à part.

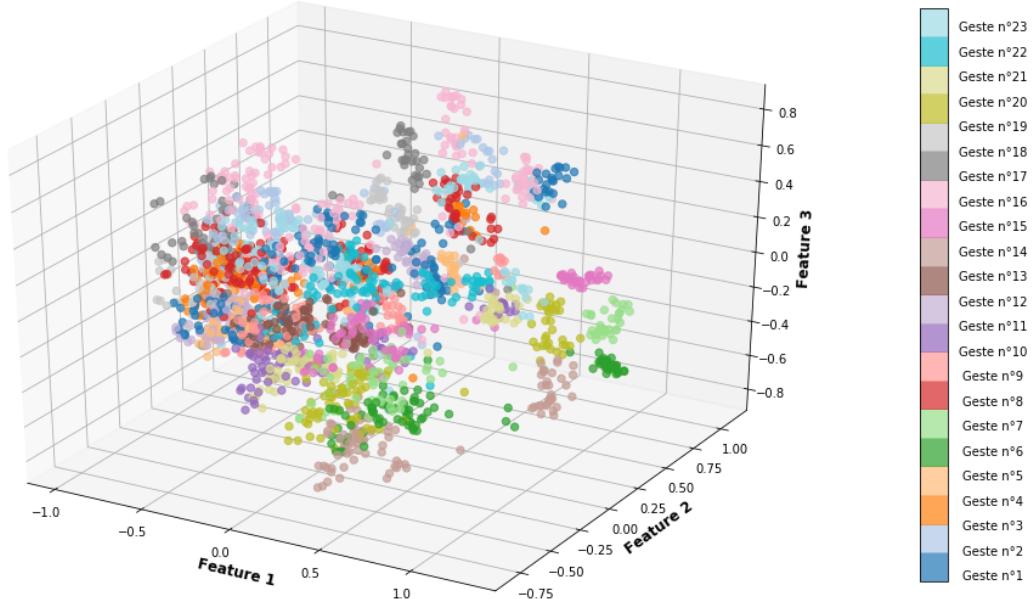


Figure 5 – Visualisation de toute la base

Ici, aucune classe ne se détache seule; mais tout en haut et tout en bas du cube (donc surtout selon l'axe de *Feature 3*), on peut peut-être distinguer les deux groupes ci-dessous. Leurs points communs montrent que la position de l'index est très discriminante : via *Feature 3*, elle suffit à faire passer un geste d'un côté à l'autre de l'espace.



Figure 6 – Groupe de similarité : index levé



Figure 7 – Groupe de similarité : index replié

Par contre, l'étrangeté est que les éléments d'une même couleur sont presque toujours éparpillés, et en particulier coupés en deux selon l'axe de *Feature 2*. Ce genre de disposition rend le problème insoluble en 3D. Il faudra beaucoup plus de dimensions pour distinguer correctement les classes.

## 1.4 Conclusion : résumé sur la base

L'étude des colonnes - descripteurs et métadonnées - a permis d'en comprendre le sens. Nous avons réussi à les interpréter comme nous le souhaitions, et obtenu une base familière et lisible pour pallier le manque initial d'information.

Colonne	Type	Interprétation
<i>Target</i>	Etiquette {1 : 24}	Métadonnée : identifiant du geste réalisé
<i>User</i>	Etiquette {1 : 8}	Métadonnée : identifiant du sujet testé
<i>Predictors 1 - 7</i>	Mesure décimale	Tracker 3D : position des mains dans l'espace
<i>Predictors 8 - 29</i>	Mesure entière	Gant Cyberglove II : angles des articulations

Tableau 4 – Description finale de la base

Dernière remarque, la visualisation des données a montré que les descriptions des gestes requéraient beaucoup plus que trois dimensions pour établir des frontières claires entre eux. Le problème n'a pas l'air d'avoir une solution évidente.

## 2 Définition. Ecrire un protocole

Le risque majeur est de confondre les types de gestes du fait de la variabilité de style due aux utilisateurs. La base est petite, or sa visualisation nous a appris qu'il existait de larges recouvrements entre les classes, et la traduction visuelle des gestes montrait bien que certains se ressemblent - même pour un humain.

Dans cette section, nous discuterons d'un protocole qui permettrait de contourner ce risque : nous détaillerons la logique de la séparation *train-test*, le choix du modèle adapté, ainsi que l'étape de *feature selection* - à la recherche des descripteurs les plus significatifs pour les gestes et donc de la dimension optimale pour les entrées.

### 2.1 Choix du mode d'évaluation

Nous cherchons à développer un modèle qui soit capable de classifier parfaitement les enregistrements. Nous utiliserons l'*accuracy* ("précision", "pourcentage de bonne classification") comme métrique d'évaluation : elle convient aux classes équilibrées.

Tout modèle testé verra sa précision calculée sur un ensemble d'évaluation choisi pour représenter la réalité du terrain. Pour notre tâche, cela veut dire que les utilisateurs en *test* doivent être inconnus du modèle entraîné. A cause du contexte (robotique sociale, usage public), nous estimons en effet qu'une telle situation est la plus réaliste : après sa formation en laboratoire, le robot sera confronté à d'autres usagers.

Dans l'article, l'entraînement de chaque modèle se fait sur sept individus. Il est validé sur l'un d'entre eux et un utilisateur séparé sert à l'évaluation. Le découpage de la base inclut donc bien une différenciation qualitative. Notre version est similaire :

- Des ensembles *train* et *test* suffiront, car la base est petite et nous n'utiliserons pas assez d'hyperparamètres pour justifier un ensemble de validation (cf. 2.2).
- Pour assurer des résultats stables, nous effectuerons une validation croisée sur plusieurs sous-ensembles de la base de test, dont voici les caractéristiques :
  - Pour une évaluation réaliste, ils doivent être composés d'utilisateurs non rencontrés en *train* - qui varieront à chaque itération de validation.
  - Pour assurer la reconnaissance, les 24 étiquettes de gestes devront toutes apparaître en *train* et en *test*. Cette condition sera remplie par défaut car les utilisateurs réalisent tous les gestes, peu importe combien on en isole.

Les normalisations des ensembles de *train* et de *test* devront se faire séparément à chaque itération de validation afin de conserver le secret sur les distributions de *test*.

## 2.2 Choix du classifieur utilisé

Il nous faut un paradigme supervisé qui permette d'apprendre un modèle à plusieurs classes. Comme le geste ne dépend pas du temps, l'information tient dans un vecteur et n'a pas à être condensée ou convoluée de quelque façon que ce soit. Il nous semble par ailleurs inutile (même si l'équipe de recherche l'a fait) d'entraîner un réseau de neurones sur une si petite base.

Le modèle doit être le plus léger possible pour les raisons détaillées en introduction. Pour aller au plus simple et limiter par défaut la consommation de mémoire, nous le choisirons parmi des classificateurs basiques que nous connaissons bien :

- Les arbres d'une Random Forest classifient les gestes au fil des *features* prises en compte. La logique est intuitive : certaines classes de gestes ne diffèrent par exemple que par la position d'un seul doigt.
- Un Support Vector Classifier (SVC) étudie la distance relative entre les gestes, toutes *features* confondues. C'est un modèle très classique.
- La régression logistique avec régularisation LASSO contient sa propre sélection de paramètres. Elle pourra éventuellement compléter la nôtre.

L'utilisation de ces modèles en baseline et leur évaluation, en page 13, nous permettra de raffiner cette liste ultérieurement.

## 2.3 Choix des dimensions d'entrée

Pour compléter le protocole, il nous reste la *feature selection*. Nous pensons que le grand nombre de dimensions initiales exigerait raisonnablement de se séparer de quelques-unes. De fait, le modèle appris doit être le plus léger possible pour assurer une reconnaissance instantanée, et cette réduction permet aussi d'éviter le surapprentissage ; un modèle moins nourri saura mieux généraliser.

En reconnaissance, il ne s'agit pas de décrire le mode d'être d'un geste, mais de retenir ses caractéristiques discriminantes. On souhaite donc réduire au maximum les dimensions d'entrée sans compromettre la qualité des résultats. Le plan est de filtrer d'abord les *features* de façon supervisée, puis de combiner celles qui restent via une PCA (Principal Component Analysis). On détaille ici ces deux étapes.

*Note : en situation, tout cela devra se faire après la séparation train-test. Sélectionner nos features sur la base de tous les échantillons invaliderait les performances du modèle en l'informant au sujet de l'ensemble de test ; or les analyses menées ci-dessous se font sur toute la base, faute de découpage fixé, et ne sont donc que des illustrations a priori.*

### 2.3.1 Suppression de *features* : filtrage

Connaissant la sémantique des colonnes de *Predictors*, nous pouvons nous prononcer sur leur utilité. Par exemple, nous affirmons que les *positions absolues* des mains pour un *geste statique* n'ont aucun intérêt. Nous le confirmons via un filtrage supervisé qui étudie la corrélation entre les *features* et les étiquettes : selon notre hypothèse, les premières colonnes du Tracker devraient montrer la même corrélation avec tous les gestes. En voici une preuve visuelle si on prend l'ensemble de la base :

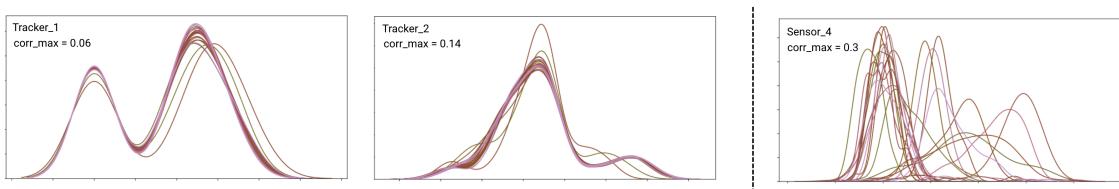


Figure 8 – Corrélation (*a priori*) des étiquettes avec *Tracker1* et *2* et comparaison au cas de *Sensor4*

Chaque graphique correspond à un seul descripteur, et montre des distributions de densité qui représentent sa corrélation avec les étiquettes (une courbe par étiquette).

L'affichage est allégé des chiffres, car le fait que les courbes se *superposent* sur les deux premiers graphiques suffit à montrer que la localisation des mains n'est pas une *feature* discriminante pour les types de gestes. Par contre, les données issues d'un capteur du gant (*Sensor4*) montrent une distribution très diverse des corrélations, on a donc tout intérêt à conserver cette information.

Lors de la mise en oeuvre, nous recommencerons ce filtrage sur chaque ensemble de train. Les *features* qui montreraient comme ici des corrélations similaires avec tous les gestes seront écartées. Notons que pour automatiser l'étape, nous n'utiliserons pas une visualisation comme ci-dessus mais une fonction *sklearn* dédiée.

### 2.3.2 Combinaison de *features* : PCA

Nous allons combiner le reste des *features* par PCA. Une PCA suppose l'existence d'une relation linéaire entre les éléments à combiner ; c'est une hypothèse forte, mais nous pensons qu'elle s'applique aux *features* issues du gant. Les capteurs au niveau des doigts nous semblent trop denses, avec des *outputs* probablement redondants, et ce même s'ils sont séparément assez informatifs pour survivre au filtrage supervisé.

L'intuition est validée par des calculs de corrélation entre les *features*.

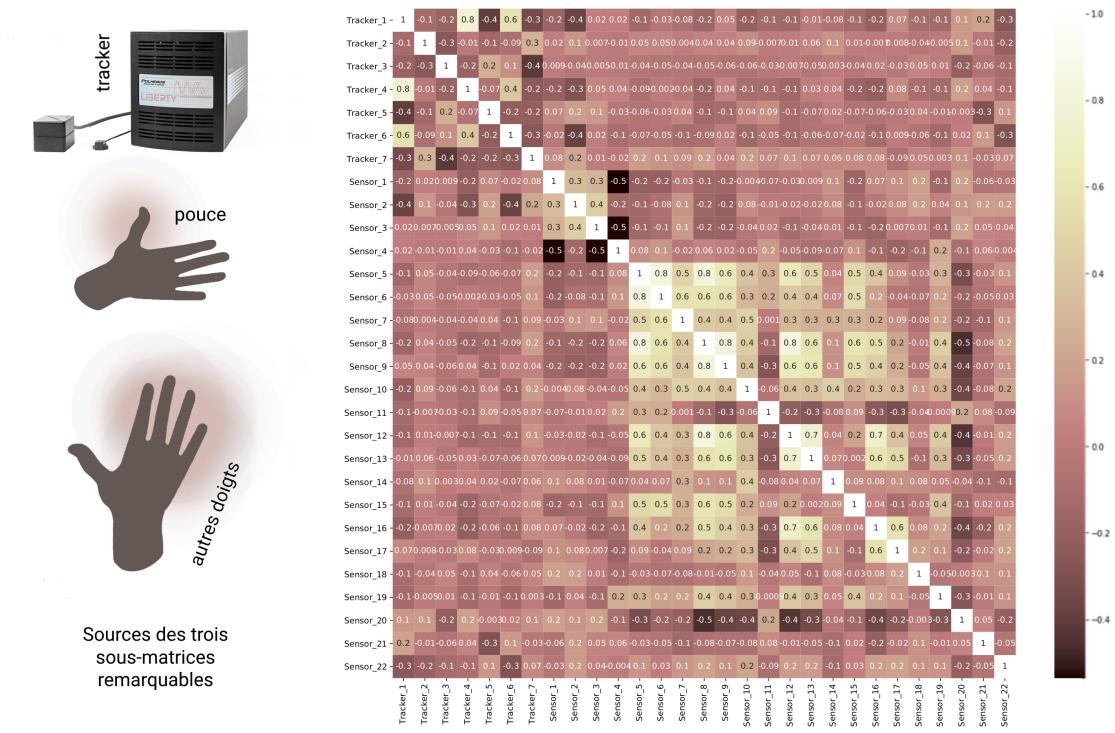


Figure 9 – Matrice de corrélation (*a priori*) des *features*

Il est vrai qu'il n'y a aucune corrélation supérieure à 80%, ce qui veut dire que le choix des capteurs était quand même fait de sorte à éviter les redondances.

Cependant, la couleur claire sur une large zone indique bien une forte corrélation entre les capteurs du gant. On peut même observer des coefficients de corrélation maximaux pour les capteurs situés sur la même phalange ou le long d'un doigt - dont ceux du pouce qui sont particulièrement liés et séparés des autres. Pendant ce temps, la sous-matrice qui met en rapport les mesures du tracker est beaucoup moins nette et laisse conclure à leur indépendance. Qu'en déduire ?

- Les *features* du Tracker n'appellent effectivement aucune transformation. La matrice ne montre aucune corrélation linéaire significative entre elles.
- En revanche, une piste concerne les mesures correspondant à la même phalange ou au même doigt dans le gant. Elles pourraient bien être combinées - pour fabriquer des *features* moins nombreuses, mais de sémantique fixée.
  - Pourtant, après réflexion, ce n'est pas recommandé. Le problème est le contexte : la lecture "pure" d'un geste selon les doigts ou les phalanges n'est pas possible, car il existe plusieurs cas où *un seul* doigt est replié. On ne peut donc pas faire des PCA selon *l'un ou l'autre* de ces deux axes.

Voilà pourquoi il semble seulement envisageable de lancer une PCA à l'échelle du gant entier, en lui laissant le soin de créer automatiquement des groupements cohérents. Par rapport à une PCA sur la totalité des *Predictors*, le fait que les *features* recombinées proviennent d'une seule source permettra de maintenir plus d'interprétabilité.

## 2.4 Conclusion : proposition de protocole

Ces explorations exhaustives nous ont aidées à rédiger notre protocole final, avec la problématique :

### Protocole expérimental - Classification supervisée en robotique sociale : reconnaissance de 24 gestes sur une base à 29 features et 8 utilisateurs

1. Mettre en place une validation croisée.  
Isoler les données d'utilisateurs différents en *train* et en *test*.
2. A l'intérieur de chaque itération de validation :
  - (a) Normaliser séparément les données de *train* et de *test*.  
C'est plus drastique qu'une pré-normalisation sur toute la base, et plus honnête : aucun indice n'est donné en entrée sur la distribution de *test*.
  - (b) Alléger leurs dimensions.  
Supprimer les *features* jugés non discriminantes en s'appuyant sur un filtre supervisé en *train*. Appliquer le même sur l'ensemble de test.
    - *Latitude d'action : choix du seuil de suppression*
  - (c) Continuer l'allègement.  
Combiner spécifiquement les caractères restants pour le gant via une PCA sur l'ensemble de *train*. Appliquer la même transformation sur l'ensemble de *test*.
    - *Latitude d'action : paramétrage de la PCA*
  - (d) Utiliser des techniques de classification supervisées pour lancer la reconnaissance des gestes.
    - *Latitude d'action : choix du modèle, paramétrage*
3. Evaluer le modèle et rechercher le paramétrage optimal.

#### On rappelle les critères d'évaluation :

- le score du modèle, car le ratio de reconnaissance doit être proche de 100 % ;
- la rapidité de réponse, car la reconnaissance doit se faire en direct ;
- et l'économie de mémoire, car la tâche ne doit pas accaparer le robot.  
Ce critère sera déjà rempli par défaut du fait de la très grande simplicité des modèles choisis ; on en accentuera l'effet avec la *feature selection*.

Ce protocole devrait nous emmener vers un modèle général et fiable au possible.

## 3 Contextualisation. Etablir des baselines

Cependant, il n'y a rien d'absolu pour commenter sa justesse. Tout dépend de ce qu'il est *possible* de faire : on cherche donc une borne minimale pour les performances jugées acceptables. Celui que l'équipe a suivi n'étant pas détaillé, les repères

chiffrés de l'article sont inutilisables. Il va falloir créer les nôtres via des modèles *baseline* et utiliser ces protocoles simplistes comme point de comparaison.

Nous avons exigé plus haut un score de reconnaissance proche de 100%. Notons que si les performances des *baselines* remplissent ce critère, la dépense d'énergie qu'on dédierait à faire fonctionner un modèle plus complexe ne serait pas rentable ; au-delà de leur intérêt comparatif, les *baselines* servent donc aussi à réévaluer le degré de nécessité de chaque étape du protocole initial.

Sur cette section, l'optimisation n'est pas un objectif en soi. Pour évaluer les modèles, on priorise la suppression du biais dans la sélection des exemples, d'où le choix d'une validation croisée sur la *totalité* de la base.

### 3.1 Modèles simplistes

#### 3.1.1 Classification aléatoire

Le Random Classifier choisit un label au hasard, indépendamment des *features*. La distribution des classes étant équilibrée, il a raison 4% du temps ( $\frac{1}{24} = 0.04$ ). Tout modèle qui dépasse cette performance sera jugé "intelligent".

#### 3.1.2 Classification monodimensionnelle

Certaines *features* peuvent décrire à elles seules les dynamiques inhérentes aux données (cf. *Titanic Challenge* sur Kaggle). On mesure la capacité prédictive de chacune pour retenir celle qui, seule, mène au meilleur score. Un tel modèle étant construit dans un espace unidimensionnel, il ne dépend pas du type de classifieur.

Avec un KNN, l'*accuracy* maximale est atteinte pour la *feature* 'Sensor4' qui correspond à un capteur du gant, la phalange à la base du pouce. Le score de bonne classification est de 16%. La position du pouce est donc extrêmement significative puisque sa seule prise en compte quadruple le score du modèle aléatoire.

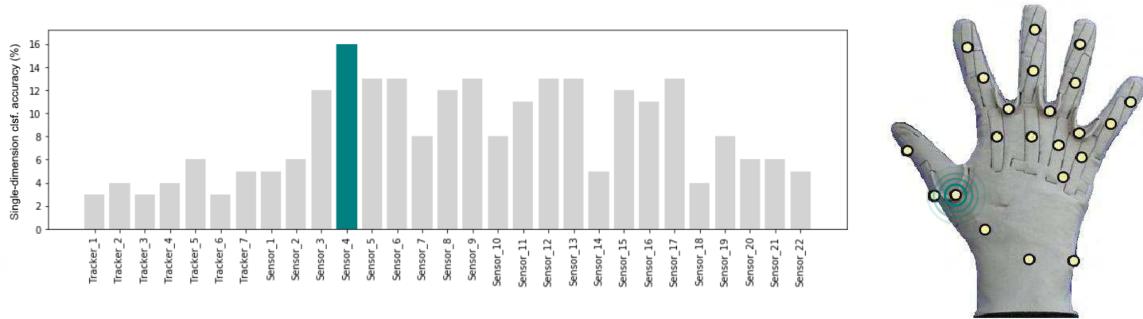


Figure 10 – Choix de la variable la plus significative et localisation du capteur concerné

Cependant, elle est très loin de permettre une reconnaissance correcte (proche de 100%) pour un robot. On n'est donc pas dans un cas type Titanic-Challenge : un modèle basique à une variable ne suffit pas. Un geste implique un certain nombre d'articulations avec trop de degrés de liberté pour tout résumer en une.

### 3.1.3 Classification en pleine dimensionalité

On déduit de ce qui précède que la prise en compte de plusieurs dimensions est importante. On peut essayer d'en inclure davantage.

Afin de garder des modèles *baseline* simples et interprétables, nous n'effectuerons cependant aucune forme de *feature engineering*. Nous considérerons les 29 caractéristiques des gestes à la fois en appliquant uniquement une normalisation.

De fait, le choix orienté des *features* prises en compte faisait tout l'intérêt du protocole proposé la semaine passée ; il est bon d'essayer de supprimer cette étape pour juger objectivement de sa nécessité.

Dans un espace multidimensionnel, il devient essentiel de choisir le modèle de classification. Comme baseline, il doit aller au plus simple pour la tâche considérée et avoir des chances raisonnables de renvoyer de bons résultats sans *fine-tuning* : on choisit donc des classifieurs dotés de peu d'hyperparamètres.

- Le K-Nearest Neighbours (KNN) étiquette un point à partir de sa distance à ses K voisins. Il ne dépend que de ce K : c'est parfait.
- La régression LASSO comporte une phase de régularisation intégrée ; sa présence par défaut permettra d'obtenir un score remarquable sans effort.
- L'arbre de décision est très basique, mais nous savons qu'il peut être combiné en forêts avec de très bons résultats (cf. section 4.2.2).

Avec leurs paramètres par défaut, les modèles sklearn correspondants renvoient :

Modèle	Précision (%)	Variance
K-Nearest Neighbours	69	0,0288
Régression LASSO	81	0,0062
Arbre de décision	72	0,0076

Tableau 5 – Résultats des modèles baseline par défaut

Ces résultats sont imparfaits mais bien meilleurs que dans les deux cas précédents. Le classifieur KNN est instable. Sa variance élevée montre une tendance au surapprentissage : son score dépend de chaque itération de validation. Pourtant, le score n'est pas satisfaisant, le biais est donc lui aussi élevé.

L'arbre de décision suit comme solution médiane ; et avec un score de 81% et une variance quasi-nulle, la régression LASSO est le modèle le plus prometteur.

## 3.2 Améliorations des baselines

Les méthodes présentées ci-dessous permettraient d'obtenir beaucoup mieux avec très peu d'investissement. On considère donc qu'il s'agit toujours de baselines.

### 3.2.1 Hyperparameters tuning

Les hyperparamètres sont parfois la clef d'une amélioration conséquente.

Concernant le KNN, un *GridSearch* sur K (le nombre de voisins considérés) n'apporte rien : la variance est trop haute pour qu'il ait un impact.

Concernant la régression LASSO, le paramètre de régularisation C est un peu plus utile. La valeur optimale était déjà la valeur par défaut, soit 0.1.

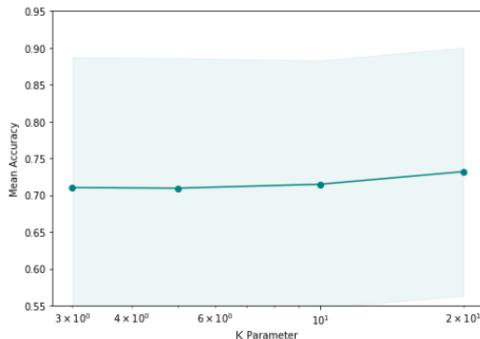


Figure 11 – Impact de K sur le score du KNN

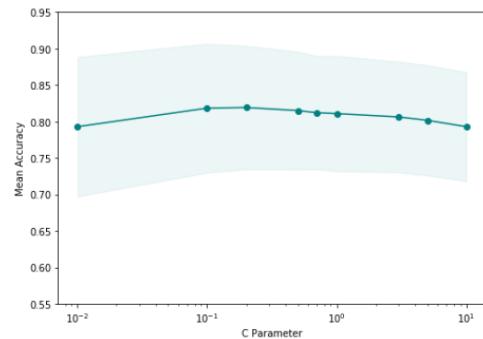


Figure 12 – Impact de C sur le score du LASSO

Cette expérience est informative malgré l'absence d'amélioration. Rappelons que la régularisation du LASSO correspond à une *feature selection* automatique : le fait que son score dépasse toujours celui du KNN prouve l'importance de cette étape.

### 3.2.2 Méthodes d'ensemble

En combinant des modèles, les méthodes d'ensemble permettent d'éradiquer sous-apprentissage et sur-apprentissage. Les méthodes par moyennage et par *boosting* sont adaptées à chaque aspect du problème - réduire soit la variance, soit le biais.

- L'instabilité du KNN appelle un *bagging* qui lisse la variance en moyennant les prédictions. Mais il n'ajoute que 2% d'*accuracy* : on reste loin des autres modèles. Il est possible que la notion de "distance" entre les points à l'oeuvre dans le KNN soit en fait inadaptée à des gestes ; ce n'est pas la *baseline* la plus fiable.
- Le score du LASSO a une marge de 19 points par rapport à la cible de 100%. On lui préfère donc l'algorithme de *boosting* XGBoost qui inclut aussi une régularisation et se concentre sur les points mal classifiés à chaque itération. Le score reste à 81%, certaines régions de l'espace sont donc simplement trop bruitées.
- On essaie également une *Random Forest*. Elle correspond à un *bagging* d'arbres, avec une protection contre le risque de créer des forêts de clones : chaque arbre considère seulement une sous-partie des *features*. Ici, on atteint 81% avec dix arbres contre 72% de bonne classification pour un arbre seul. C'est satisfaisant.

Modèle	Précision (%)	Variance
Random Forest	81	0,0051
Bagging de KNNs	71	0,03
Boosting (XGBoost)	81	0,0054

Tableau 6 – Résultats des méthodes d'ensemble, validation croisée à cinq itérations

Le score final ne dépasse jamais 81%. Quelles sont les erreurs commises?

XGBoost, qui atteint ce score maximal, produit la matrice de confusion suivante (il s'agit précisément de la somme des matrices de confusion sur les folds de test). On a ajouté une visualisation des erreurs : un humain peut justifier la plupart, car la différence est minime, mais certaines comme la confusion 7 / 24 semblent évitables.

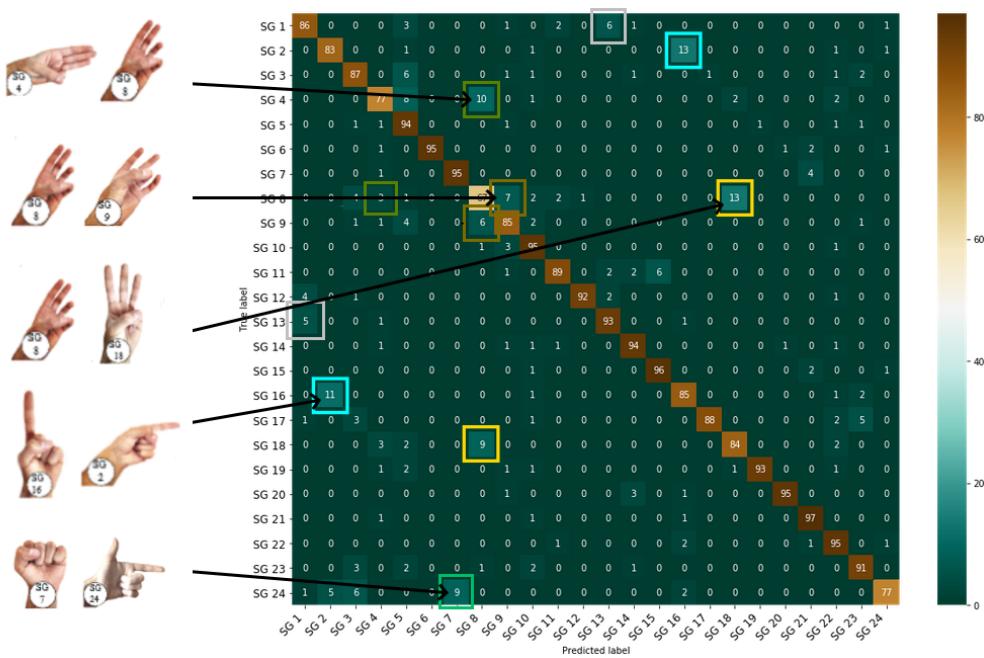


Figure 13 – Matrice de confusion pour XGBoost en test

Une seconde matrice de confusion a été générée pour la Random Forest (disponible ici) qui parvenait au même score de 81%. Les erreurs sont similaires, ce qui semble indiquer que des zones précises de l'espace posent problème. Dans la section suivante, une *feature selection* intelligente devrait réduire l'incertitude à leur sujet.

### 3.3 Remarque méthodologique

Les scores données ici dépendent d'un code personnel. Les résultats en validation croisée sklearn sont bien meilleurs. L'écart s'explique : rappelons que nous renormalisons les données dans chaque itération de la validation croisée, ce qui rend le problème plus difficile ; nous faisons en sorte que la distribution de l'échantillon de test soit toujours découverte à l'aveugle. Il n'est pas dit que cette précaution est toujours prise et elle n'existe pas par défaut dans sklearn.

En illustration, nous proposons les courbes d'apprentissage en page suivante qui culminent toutes à plus de 85% d'*accuracy*. Bien que le score final diffère, la hiérarchie de performance des modèles correspond quand même à nos observations.

Dans l'optique de surveiller l'apprentissage, nous réutiliserons néanmoins notre propre code de validation croisée pour appliquer le protocole. Le maximum de 81% obtenu avec en baseline est donc la seule référence valable.

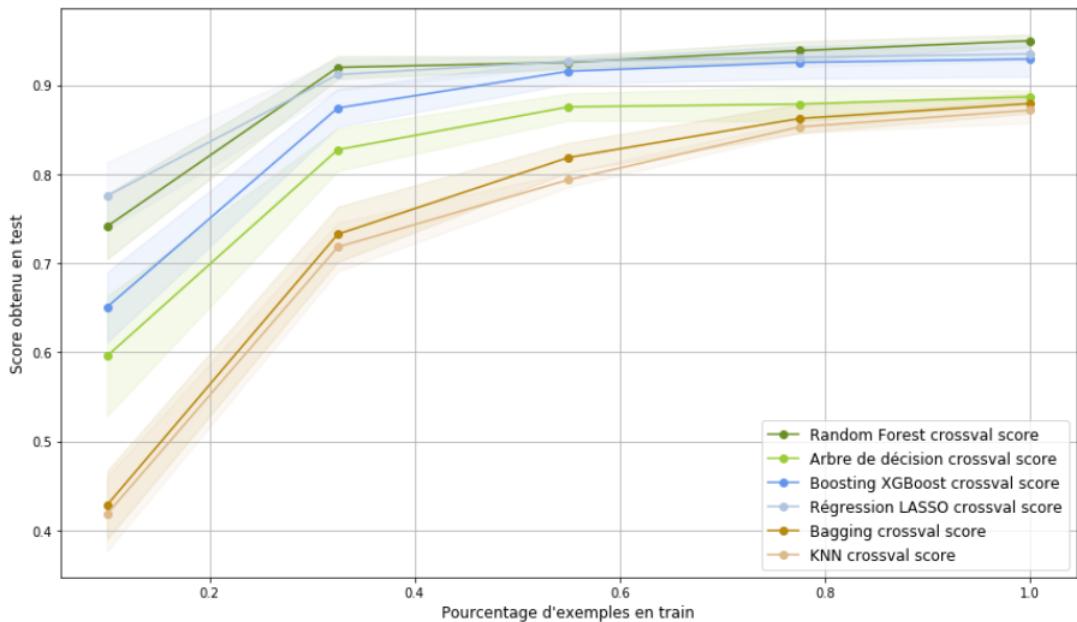


Figure 14 – Visualisation des courbes d'apprentissage pour les modèles sklearn en baseline

### 3.4 Conclusion : discussion des objectifs

Selon ce que nous renvoient les *baselines*, il faudrait qu'un modèle plus avancé dépasse les 81% de prédictions justes en battant XGBoost. Ce score reste faible par rapport à ce qui est souvent présenté dans la littérature ; cela s'explique.

- Essentiellement, aucune *feature selection* n'a été entreprise.
- Nous renormalisons les données à chaque itération de la validation croisée, ce qui met les modèles en difficulté. Ils découvrent vraiment la distribution de test à l'aveugle. Nous l'avions annoncé dans le premier protocole, et par souci de réalisme, nous persistons dans cette voie.
- Notons que nous avions obtenu des scores plus élevés en considérant des sous-parties des données. La baisse nette dès lors qu'on inclut toute la base est due à la multiplicité des gestes et des styles des utilisateurs : plus les modèles sont exposés à ces données bruitées, moins ils sont immédiatement capables de généraliser.

Nous pensons que notre protocole aura un impact positif en assurant

1. un choix de caractéristiques éclairé, capable de réduire intelligemment la dimension des données et donc le bruit dans les dynamiques extraites,
2. une séparation *train-test* bien orientée dans la validation croisée, de sorte à interpréter séparément des données de sémantique différente (utilisateurs familiers ou nouveaux arrivants),
3. une optimisation des hyperparamètres bien menée, qui tient compte séparément des résultats sur ces deux ensembles et n'en est que plus ciblée.

## 4 Application. Mise en oeuvre du protocole

Cette section présente enfin les résultats. Les modèles retenus sont les mêmes qu'annoncé en page 7, soit une RandomForest, une régression LASSO et un SVC. Il nous semble que les autres modèles essayés en *baseline* (KNN, XGBoost...) n'ont pas apporté d'amélioration particulière par rapport à ceux-ci. Nous justifions ce choix :

- La Random Forest est censée donner les meilleurs résultats selon l'article et semblait très satisfaisante en *baseline*.
- La régression LASSO était aussi efficace et sa régularisation intégrée nous intéressait toujours.
- Le classifieur SVC ne pouvait pas être considéré comme une *baseline* du fait du grand nombre de paramètres ; c'est maintenant l'occasion de l'essayer.

L'objectif est cette fois de bien mener l'optimisation pour donner nos scores finaux sur la tâche. Malgré la difficulté apportée par nos normalisations indépendantes, est-on capable de battre le score annoncé de l'article - 95.6% avec une Random Forest ?

### 4.1 Tâche principale

Pour les trois modèles, on reprend d'abord le protocole de la page 10 : on classe les gestes pour tous les utilisateurs.

1. Nous effectuons une validation croisée manuelle où les ensembles de *test* sont à chaque fois composés de deux utilisateurs sur les huit. Cela permet de constater la généralisation, mais offre l'avantage d'aller plus vite que le *leave-one-[user]-out* (quatre combinaisons possibles, soit quatre itérations seulement) tout en gardant l'ensemble de test minoritaire par rapport au *train*.
2. A l'intérieur, on utilise des fonctions de sklearn :
  - (a) La normalisation des données de *train* est standard (diviser par l'écart-type, retirer la moyenne). On applique la même aux données de *test*.
  - (b) On applique le filtrage supervisé à l'ensemble de *train* en réglant la proportion de *features* conservées. Les données de *test* sont traitées de même.
  - (c) On applique une PCA sur ce qui reste des caractères du gant en *train*. On applique la même transformation au set de *test*.
  - (d) On lance un classifieur.
3. Sortie de la validation croisée et lecture du score.

#### 4.1.1 Etude approfondie de la *feature selection*

La clef de l'amélioration par rapport aux *baselines* (81% d'*accuracy*) réside probablement dans la sélection des descripteurs par filtrage supervisé puis PCA.

Nous voudrions observer l'impact de chacune de ces méthodes - seule - sur le score final. Cependant, au vu du mode de validation différent, les scores *baseline* ne peuvent pas servir de référence. L'ajout d'un split *train-test* avant la validation croisée (absent de l'évaluation des *baselines*) aide les modèles en limitant la variété des utilisateurs.

On constate ci-contre que les scores dans ce paradigme sont déjà plus élevés que 81%, même sans *feature selection*. Un paramétrage de filtrage ou PCA sera donc utile seulement s'il permet de dépasser ces scores de base.

Modèle	Précision	Variance
RandomForest	91.0%	0.0014
SVC	89.0%	0,0028
LASSO	85.0%	0,0033

Tableau 7 – Résultats sans *feature selection*

La RandomForest - la méthode privilégiée par l'équipe - tient déjà la première place, mais avec un score plus faible que prévu (91%). Est-ce améliorable juste avec l'une ou l'autre de ces techniques ?

Pour le même *random seed* et tous paramètres égaux par ailleurs, les trois modèles sont testés ; dans un cas, on applique le filtrage supervisé dont le percentile de *features* conservées varie de 5 (filtrage quasi-total) à 100 (aucun filtrage) : dans l'autre, la PCA est appliquée seule pour conserver entre 50 et 99.9% de variance expliquée.

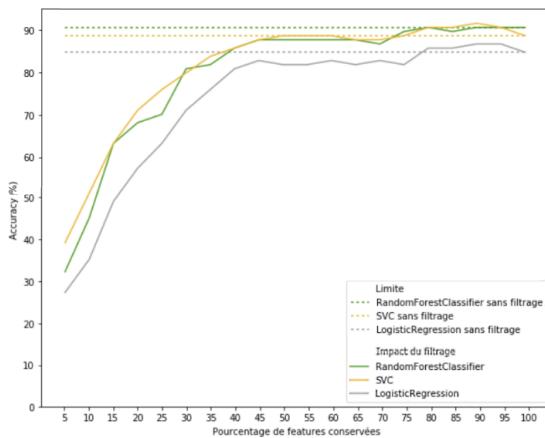


Figure 15 – Impact isolé du degré de filtrage

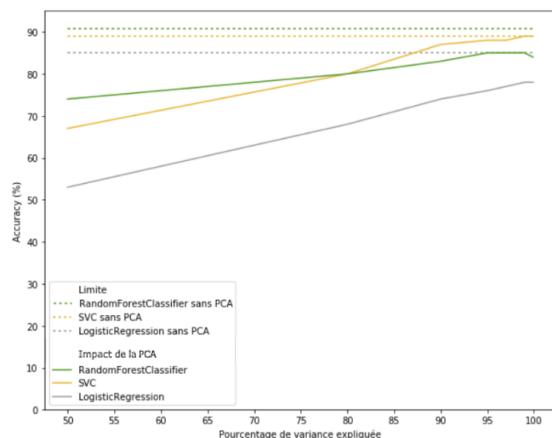


Figure 16 – Impact isolé de la PCA

Dans tous les cas, les résultats post-filtrage sont à peu près les mêmes que l'on garde 45 ou 100% des *features*. Cela signifie qu'une majorité d'entre elles ne jouent pas un rôle majeur. Des essais manuels montrent donc parfois un score exceptionnel pour un filtrage à 70 ou 80% ; la fixation du *random seed* pour les besoins de la comparaison nous empêche de constater cette variabilité ici.

En particulier pour le SVC et le LASSO, un filtrage léger (retenir 90% des *features*) est associé à un pic de performance. Il donne de meilleurs résultats qu'une prise en compte de toutes les dimensions originales. Pour la RandomForest, les performances avec filtrage sont toujours inférieures à la référence, car elle effectue déjà elle-même sa sélection de paramètres ; sa réaction négative à cette transformation forcée fait qu'elle se laisse légèrement dépasser par le SVC.

Quant à la PCA, c'est en son absence totale que l'on obtient les meilleurs résultats. Effectuée seule, elle réduit largement le score de tous les modèles, même en conservant 99.9% de variance expliquée ; et la pente de la courbe est trop forte pour considérer un *tradeoff* entre réduction de dimensionnalité et optimisation du score. C'est donc seulement la combinaison de la PCA avec le reste qui sera utile aux modèles.

#### 4.1.2 GridSearch et détection des paramètres optimaux

Une fois cela compris, on lance un GridSearch extensif pour connaître le paramétrage optimal sur toutes les étapes réunies. Les paramètres testés reviennent à effectuer un produit scalaire sur deux ensembles, concernant

- côté classifieurs : nombre d'arbres (5 à 50) et profondeur (20, 50, sans limite) de la RandomForest, régularisation C (0.001 à 10) du SVC pour tous les types de kernel, régularisation C (0.001 à 10) du LASSO
- côté *feature selection* : nombre de *features* conservées par le filtrage (5 à 100%), variance expliquée par la PCA (50 à 99.9% ou absence de transformation).

Le tableau suivant résume le top 5 de toutes les combinaisons étudiées :

Modèle	Paramètres	Degré filtrage	Sévérité PCA	Précision	Variance
SVC	kernel RBF, C=10	Maintien 90%	V. exp. 95%	92.0%	0.00115
SVC	kernel RBF, C=10	Maintien 90%	V. exp. 97%	92.0%	0.00116
SVC	kernel RBF, C=10	Maintien 90%	V. exp. 99%	92.0%	0.00119
SVC	kernel RBF, C=1	Maintien 90%	V. exp. 99%	92.0%	0.00121
SVC	kernel RBF, C=10	Maintien 80%	V. exp. 99%	91.0%	0.00144

Tableau 8 – Choix du paramétrage optimal pour le modèle final

La différence majeure entre les cinq lignes est le paramétrage de la PCA (sans impact sur le score ni sur la variance, qui est faible) : il s'agit toujours de SVC avec un noyau gaussien. La constante de régularisation C varie sans impact non plus. La Random Forest est détrônée et le score maximal est apparemment de **92% de précision**.

Le filtrage semble décisif pour ce modèle (maintien de 90% des *features* comme annoncé par les courbes plus haut). Avec la dernière ligne du tableau, il apparaît cependant qu'on peut conserver moins de *features* (80%) avec un score qui demeure au même niveau (91% vs. 92% d'*accuracy*). Cela illustre notre propos selon lequel un certain nombre d'entre elles sont superflues.

#### 4.1.3 Etude du modèle optimal

On retient la toute première ligne du tableau : un SVC à noyau gaussien, avec 90% de *features* conservées lors du filtrage, plus une PCA sur les *features* du gant à 95% de variance expliquée. Qu'est-ce qui fait son optimalité ?

Premièrement, à toutes les itérations de validation croisée, ce modèle filtre automatiquement les *features* *Tracker1*, *Tracker2* et *Sensor18*. Leurs profils de corrélation avec les 24 étiquettes sont donnés ci-dessous pour l'une de ces itérations :

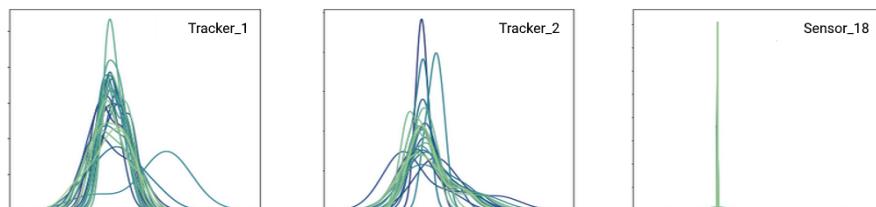


Figure 17 – Profils de corrélation des *features* systématiquement supprimées par le modèle optimal

Leur suppression est justifiée par le peu de variabilité constatée. Nous avions déjà jugé *Tracker1* et *2* inutiles (position de la main dans l'espace) et *Sensor18* représente l'angle de la dernière phalange du petit doigt. On s'en sépare sans regret.

Le modèle agit ensuite sur la combinaison des *features* du gant et la PCA se débarrasse encore de sept *features* sur les 22.

Nous souhaitons étudier l'impact de ces transformations sur la matrice de confusion du modèle. En comparant avec la Figure 13 pour la *baseline XGBoost*, celle-ci est débarrassée de plusieurs points de confusion. Voir ce lien pour l'affichage côte à côté.

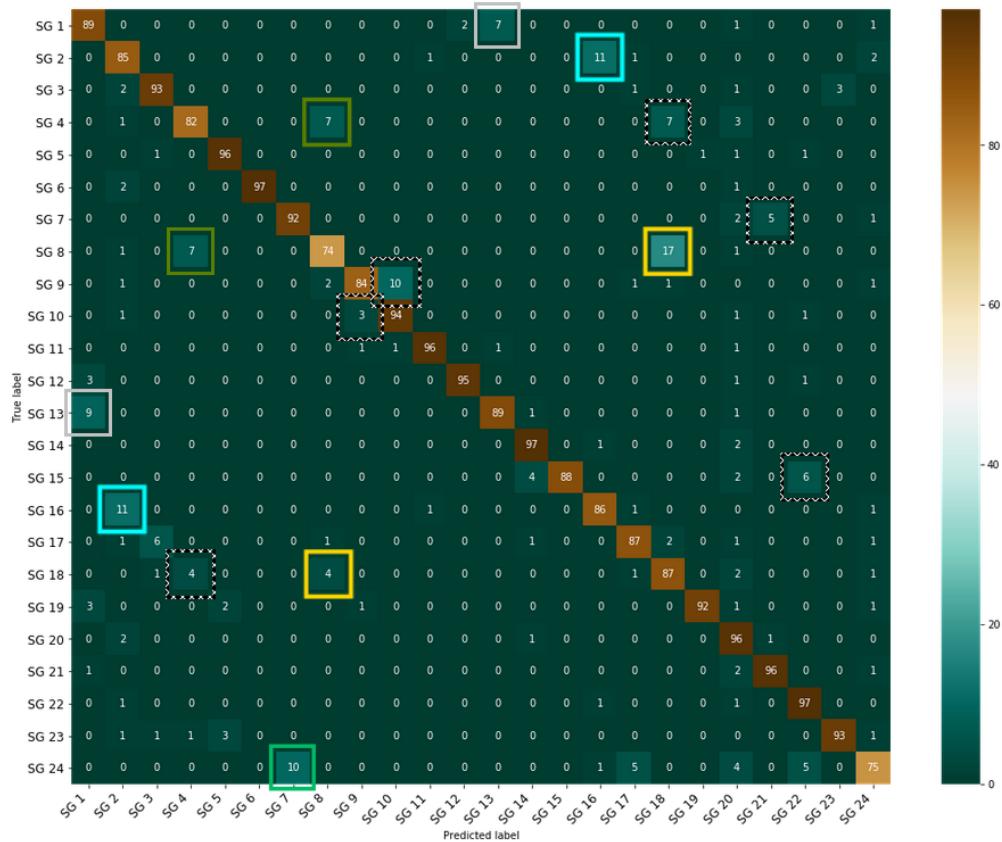


Figure 18 – Matrice de confusion du classifieur optimal général en test

On a repris le même code couleur. La différenciation des gestes 2 / 16 et 8 / 18 est toujours difficile, le problème entre 7 et 24 existe encore, mais on ne constate plus aucune confusion entre 11 et 15 par exemple. D'autres sources d'erreur sont apparues (encadré pointillé) mais ont proportionnellement moins d'importance.

Par ailleurs, la réponse est toujours instantanée, le critère de rapidité est rempli.

**Les modifications automatiques ont donc eu un effet satisfaisant par rapport au cas de base, et nous avons exposé et justifié les raisons pour lesquelles elles ont été choisies. Avec ces algorithmes, nous avons fait au mieux. Nous en avons donc fini avec la tâche principale avec un score de 92%.**

## 4.2 Extension : classification monoutilisateur

Pour s'assurer que le modèle est adéquat, on aimerait savoir si de tels hyperparamètres permettent de reconnaître à 100% un utilisateur seul. Le robot ainsi préparé peut-il fonctionner comme assistant personnel?

Nous n'allons pas *optimiser* le modèle pour cela : ceci n'est qu'un test supplémentaire pour mieux comprendre la classification déjà faite. Nous n'espérons pas faire mieux qu'avec du *fine-tuning*, comme par exemple sur un réseau de neurones.

On effectue donc une validation croisée sur les gestes de chaque utilisateur, séparément. Le tableau des résultats ci-dessous montre qu'ils varient en fonction de la personne concernée.

Les résultats sont globalement très bons et supérieurs au score général, mais celui de User1 est aberrant. Pour le comprendre, on étudie sa matrice de confusion en la comparant aux autres.

Utilisateur	Précision	Variance
1	70.0%	0.323157
2	94.0%	0.063311
3	97.0%	0.034583
4	99.0%	0.021651
5	96.0%	0.013819
6	93.0%	0.052705
7	92.0%	0.027639
8	96.0%	0.014434

Tableau 9 – Précision obtenue sur un utilisateur

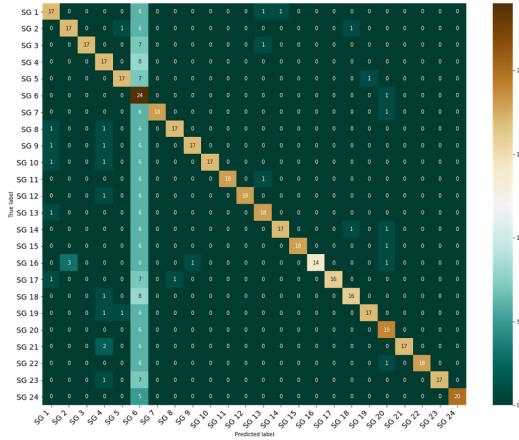


Figure 19 – Matrice de confusion pour User1

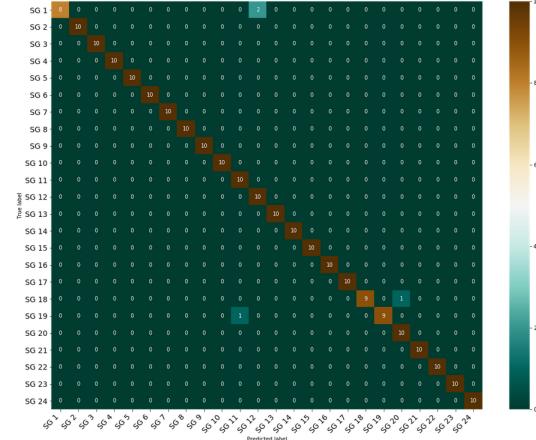


Figure 20 – Matrice de confusion pour User4

Tous les gestes de User1 sont en fait confondus avec le pouce levé (geste n°6). Ce n'est le cas pour aucun autre utilisateur. Comme c'est un geste très évident, nous pensons que c'est une erreur expérimentale qui explique cela, et elle impacte probablement le score trouvé en page précédente pour le cas général.

Outre cela, les scores par utilisateur sont satisfaisants. Une optimisation séparée permettrait certainement de les améliorer dans un contexte qui le demanderait.

## 4.3 Extension : suppression du Tracker

Pour clore cette étude, nous avons pensé à une dernière chose. Le matériel utilisé pour l'expérience consiste en deux unités distinctes ; mais comme les *features* du gant restent largement majoritaires après filtrage et PCA, le Tracker est-il vraiment utile ? S'en débarrasser permettrait d'alléger l'environnement pour la commande du robot.

Nous savons qu'il existe quatre gestes où la rotation de la main est le seul facteur discriminant (2 / 16, 1 / 13). Le gant ne peut renvoyer aucune information à ce sujet.

Cependant, la matrice de confusion du modèle optimal montre que les erreurs les concernant sont déjà majoritaires en présence du Tracker. Peut-on faire sans ?

Si ce sont les seules situations de flou (4 gestes sur 24, soit un sixième du total), un compromis entre facilité d'usage et reconnaissance parfaite est envisageable. A supposer que la confusion soit totale entre ces quatre gestes, et étant donné que la répartition entre les classes est uniforme, on peut donc espérer un ordre de 83.3% ( $\frac{100}{6} = 16.7$ ) de bonne classification.

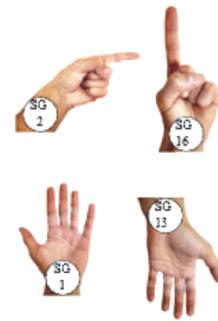


Figure 21 – Gestes indifférenciables sans la rotation des mains

Après GridSearch sur les trois classificateurs (SVC, RandomForest, LASSO), on isole un score optimal de 90%. Il est obtenu sans PCA, avec toujours un SVC à noyau gaussien et un filtrage assez drastique des *features* du gant à 70% (retrait de *Sensor* 20, 21, 22, 19, 18 et 14, des éléments associés à l'auriculaire et à la paume de la main).

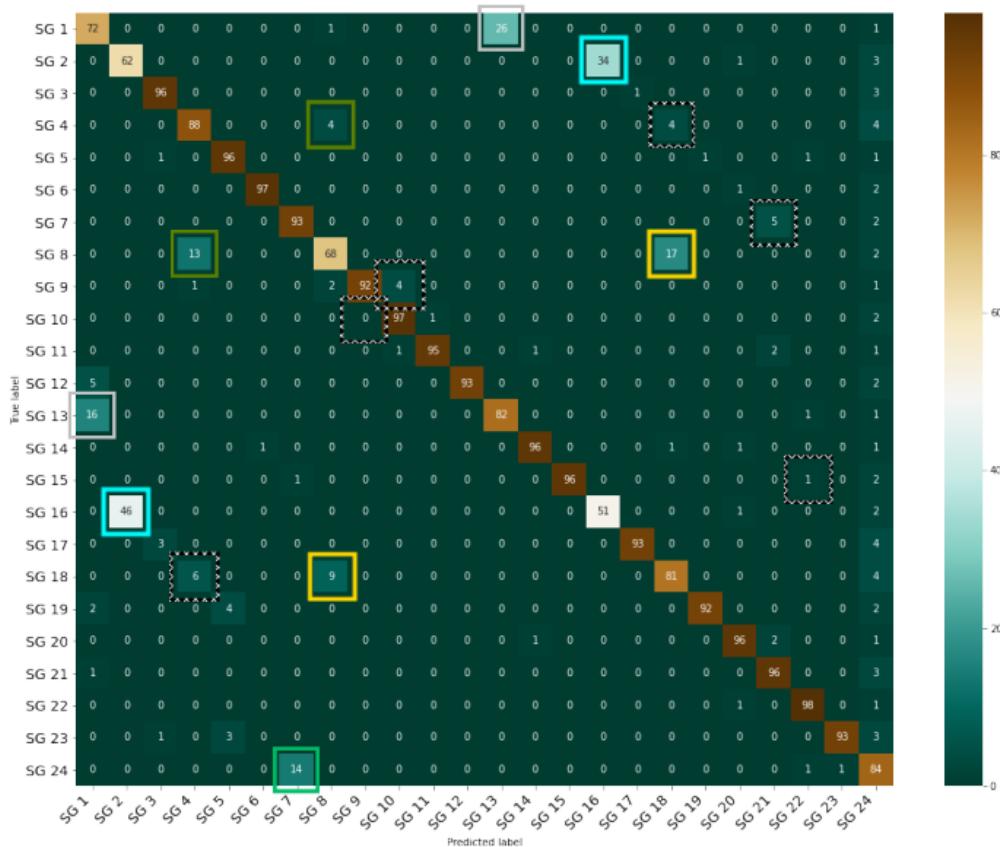


Figure 22 – Matrice de confusion du classifieur optimal sans Tracker en test

Les encadrés sur cette nouvelle matrice de confusion correspondent aux sources d'erreur du modèle général dans la Figure 18. Certaines ont été atténues, mais la confusion 2 / 16 et 1 / 13 est lourde. Permet-elle de statuer sur le sort du Tracker ?

- La proximité du score de la version sans Tracker avec le score initial rend sa suppression tout à fait envisageable. Cependant, la métrique de l'*accuracy* n'est pas forcément adaptée à ce nouveau problème pour prendre une décision.
- Même sans connaître la sémantique exacte des gestes, on se rend bien compte que la différenciation 1 / 13 et 2 / 16 peut être utile pour indiquer une direction au modèle. Ce sont des gestes quotidiens, naturels. Si l'on veut supprimer le Tracker, il faut donc peut-être envisager de remapper le dictionnaire des gestes.

Dans ce cadre, le choix se fait à la discréction de l'expérimentateur.

## Le mot de la fin

**L'essentiel de cette étude a donc porté sur le développement d'une technique adaptée pour la sélection des *features*. Dans le cadre d'une séparation *train-test* réaliste, dictée par la situation, elle a porté ses fruits en augmentant les performances des cas de base - pour atteindre 92% de précision sur la reconnaissance des gestes statiques.**

Ce modèle remplit les trois critères d'évaluation du protocole (p.10).

- 92% est un score satisfaisant étant donnée la difficulté de l'évaluation.
- La reconnaissance est instantanée et la réponse se fait *a priori* sans délai.
- Le modèle SVC utilisé est très léger, c'est une méthode simple au possible, et la *feature selection* réduit encore la consommation de mémoire.

**Peut-on alors critiquer ce modèle ? Le point négatif dans les chiffres est son score**, et nous pensons que les erreurs qu'il commet (8% de marge avec le score parfait) sont partiellement dues à celles des utilisateurs humains. Il y a tant de gestes à réaliser qu'il est très probable que leurs performances soient incertaines, a fortiori pour User1 qui a apparemment eu un problème technique conséquent.

**Cependant, nous nous intéressons aussi à ce que ce résultat implique pour les chercheurs et les utilisateurs du robot.** Pour obtenir le score optimal, nous avons ignoré certains capteurs du CyberGloveII. Mais il est impossible de les désactiver ; les données seront toutes recueillies quoi qu'il arrive. A moins de créer un autre gant, un preprocessing sera donc toujours nécessaire. Le travail d'intégration de ces données ne disparaîtra pas et on n'économise aucune étape dans l'expérience.

L'enjeu avec le plus d'impact réel était de supprimer des éléments tangibles de l'environnement expérimental, ce que nous avons étudié en tentant une suppression du Tracker. Si l'on est prêt à sacrifier 2% de bonne classification (et une certaine acceptation de la sémantique des gestes), ce peut être une piste intéressante. Cependant, toute cette étude a totalement ignoré l'existence des gestes dynamiques (DG) qui sont lus séquentiellement par le robot, et où le Tracker joue un rôle primordial ; bien que certains soient reconnaissables avec le gant seul, nous pensons que notre proposition serait totalement irrecevable dans ce cadre.