

Rapport d'avancement du groupe “5.4”

SportBot

Table des matières

SportBot	1
Membres du groupe	1
Tuteur	1
Encadrant génie logiciel	1
Résumé du sujet choisi en français (PAN1)	2
English Summary (PAN1)	3
Notes concenant le rapport	4
1. Étude d'antériorité et justification de la proposition (PAN1)	5
1.1. Description de la proposition	5
1.2. Description de l'état de l'art	6
2. Scénarios d'usage (PAN1)	8
2.1. Scénarios d'usage	8
2.2. SportBot	8
3. Architecture du projet (PAN1)	10
3.1. Schéma d'architecture	10
3.2. Diagramme de séquence	13
3.3. Interface utilisateur graphique	14
3.4. Tableau détaillé des tâches	14
4. Organisation du projet (PAN1)	18
4.1. Diagramme de planification temporel des tâches	18
4.2. Répartition des élèves par module	19
4.3. Plans de test (PAN2+)	20
4.4. Diagramme d'avancement des tâches (PAN2+)	23
Bibliographie (PAN1+)	26
5. Annexes	27
5.1. Fiche d'identité du groupe (PAN1)	27
5.2. Modifications (PAN2+)	28
5.3. Comptes Rendus de réunions	29
5.4. Synthèse 3D	36
5.5. Module Classification	39
5.6. Kinect	48
5.7. Android	56
5.8. Intégration et tests	58

SportBot

Membres du groupe

- BONNARD Florian
- VUONG Christophe
- TORCHY Axel
- BOUZAFOUR Anas
- BENJELLOUN ZAHAR Ali
- GODARD Antonin

Tuteur

- PRIEUR Christophe

Encadrant génie logiciel

- DUFOURD Jean-Claude

Résumé du sujet choisi en français (PAN1)

SportBot est une application sur smartphone qui fait office de coach sportif virtuel. En intérieur, une Kinect détecte les mouvements de l'utilisateur qui reproduit ce qui est affiché sur l'écran de son smartphone. Les mouvements captés sont alors jugés et le but est bien de le faire progresser notamment en filmant ses performances et en établissant un historique de celles-ci. De même l'autre objectif est de l'amener à aimer la pratique sportive. C'est pourquoi l'application peut préalablement avoir accès aux préférences de l'utilisateur et s'en servir pour rendre plus agréable les séances. En extérieur, l'utilisateur n'est pas jugé mais peut toujours s'entraîner et voir le bilan de ses efforts grâce au smartphone. Ainsi, il peut emmener SportBot partout.

English Summary (PAN1)

SportBot is a smartphone application that consists in a virtual sports coach. When the user is inside, a Kinect detects his movements while he replicates what is displayed on the screen of his smartphone. Captured movements are then assessed and the goal is to make him improve in particular by filming his performances and establishing a history of them. It is also meant to get him to love sports. It is why the application can access the preferences of the user before the session starts and uses it in order to make the training more enjoyable. When he is outside, the user is not assessed but he can still practice and have a feedback on his efforts with the smartphone. So, he can take SportBot everywhere.

Notes concernant le rapport

Les différentes pages du document sont rédigées en utilisant le langage AsciiDoc. Le squelette de rapport contient des exemples avec entre autre:

- des images,
- des liens,
- des équations.

La structure du rapport (parties, sections et la relation avec les différents fichiers) se trouve dans le fichier courant.

Chapter 1. Étude d'antériorité et justification de la proposition (PAN1)

1.1. Description de la proposition

Notre projet consiste en une application qui fait office de coach sportif virtuel nommé SportBot. Cette application utilise la Kinect pour analyser les mouvements gymniques de l'utilisateur et de l'encourager dans sa progression. Initialement, le robot était supposé réel et le but de celui-ci était de devenir le véritable ami de l'utilisateur au fil des séances au point de le taquiner et de se moquer gentiment de ses erreurs lors des sessions sportives. Ce robot devait avoir des fonctionnalités telles qu'il pourrait devenir envahissant. Par exemple, on avait imaginé que le robot eusse pu s'allumer tout seul et remuer l'utilisateur même s'il n'a pas très envie de faire du sport, ce afin de concevoir un vrai golem numérique. Or, l'école n'avait pas les outils nécessaires pour envisager la conception d'un tel robot qui serait cher par rapport à la plus value apportée. Après mûres réflexions, on a jugé qu'avoir accès aux goûts ou préférences de l'utilisateur sous son autorisation, programmer la séance pour lui et proposer de filmer ces performances pendant les séances en intérieur sont des opérations susceptibles de créer un lien avec l'utilisateur. Ainsi, il ne considérera pas SportBot comme un gadget, mais bien une entité assez intelligente au point de s'y attacher.

Nous avons rapidement envisagé de rendre les séances plus agréables en ajoutant à l'application une fonctionnalité qui permettra de jouer les musiques préférées lors des exercices. Nous avons retenu cette idée lors de la mise en oeuvre du scénario du projet. Le robot, à défaut d'être virtuel, pose des questions variées durant l'effort afin de créer un lien. Or, il ne saura pas en mesure d'engager une vraie conversation avec l'utilisateur. En effet, cela exigerait d'ajouter une fonctionnalité de reconnaissance vocale adaptative et complexe selon les retours que l'on a eu à la foire aux experts.

Finalement, les questions posées par le robot ne seront pas susceptibles de démarrer un dialogue, mais d'apprendre un peu plus sur l'utilisateur. De plus, il sera possible pour l'application de lancer des phrases programmées à certains instants afin d'animer encore plus la séance.

De plus, dans notre scénario nous avons aussi pensé la possibilité d'utiliser SportBot en extérieur, quoiqu'on ne pourra proposer qu'un mode de séance limité de par l'impossibilité d'installer le matériel pouvant traiter les données captées par la Kinect. On propose simplement les fonctionnalités liées au smartphone.

Lors de l'élaboration des diagrammes, nous avons pu affiner notre projet, en identifiant les modules. On a rapidement identifié nos modules de base que sont le module Kinect et le module base de données. On a pensé à quand il faut appeler la base de données et à comment limiter ces appels d'où l'inclusion du module communication client-serveur. On a aussi considérer ceci pour permettre une utilisation optimale de l'application en extérieur. Nous avons aussi pensé le stockage de certaines données sur le smartphone, notamment l'historique des séances, et les photos de celles-ci.

On s'est alors concentré sur les modules qui concerne le robot virtuel coach sportif.

Tout d'abord, nous avons donc dû réfléchir à comment générer la banque de mouvements pour les

exercices, ce qui a mené à penser une API Synthèse 3D. En effet, à partir de quelques exercices de base, on modéliserait un robot virtuel en 3D que l'on pourrait mouvoir via un logiciel de traitement 3D pour créer notre banque de mouvements qui contiendrait les erreurs possibles sur les exercices. Après consultation de l'expert en synthèse 3D, on a pris conscience qu'une telle entreprise était plus complexe que de simplement modéliser les mouvements en les faisant faire à une personne aguérie de façon répétée et de par la suite faire une moyenne des valeurs tirées de la capture de mouvements par Kinect pendant la répétition des exercices pour établir le mouvement en question. Par ce procédé, il sera aussi facile de représenter les erreurs sur le mouvement dans notre base de mouvements. Ainsi, nous nous sommes mis d'accord avec l'expert sur cette alternative.

Ensuite, on s'est demandé comment déceler à partir de notre banque de mouvements les erreurs de notre utilisateur. D'où l'introduction de la classification par k-PPV dans notre projet. Or, il a fallu consulter l'expert sur ce domaine pour se rendre compte que dans notre cadre un module plus large est nécessaire et on pourrait simplement l'appeler classification , un module qui prend en compte la spécificité des images obtenues par Kinect et la nécessité de faire du recalage temporel et spacial sur celles-ci lors de la comparaison avec les données présentes dans la banque de mouvements.

A présent, nous avons validé tous nos modules et défini entièrement notre projet.

1.2. Description de l'état de l'art

1.2.1. Un coach virtuel utilisant la Kinect

La principale fonctionnalité de la Kinect étant la détection des mouvements, l'idée de l'utiliser pour des applications *sportives* et plus seulement *ludiques* est rapidement apparue.

Plusieurs jeux sur Xbox 360 utilisent la Kinect dans une optique sportive, par exemple en proposant un coaching sportif. C'est le cas de :

- **Kinect Training** développé par *Microsoft Studios* et *Sumo Digital* ;
- **Adidas MiCoach** développé par *505 Games* ;
- **Your Shape: Fitness Evolved 2012** développé par *Ubisoft*.

Ces jeux utilisent la technologie Kinect mais sont développés sur la console de jeux, ce qui diffère du fonctionnement de **SportBot**.

1.2.2. Des coachs sportifs sur Android

Sur Android, il existe de nombreuses applications faisant office de coach sportif et très populaires, notamment concernant le *running*. On citera par exemple :

- **Runtastic Course à pied, Running, Marche** développé par *Runtastic*, une des applications les plus complètes et populaires dans ce domaine ;
- **RunKeeper** développé par *FitnessKeeper, Inc.* qui met l'accent sur l'utilisation de la fonctionnalité GPS du téléphone ;
- **Runtastic Push-Ups PRO** développé par *Runtastic*, qui accompagne l'utilisateur pour faire des

pompes en construisant un entraînement progressif.

Ces applications Android diffèrent également du fonctionnement de **SportBot** : pour celles qui proposent un accompagnement avec un entraînement sportif, ce dernier est plutôt "pragmatique" : il n'y a pas d'interaction *personnalisée* entre l'application et l'utilisateur, contrairement à ce que propose **SportBot**. De plus, il n'y a aucun moyen de vérifier si les mouvements effectués sont corrects, et *a fortiori* aucun moyen de les corriger : grâce à la Kinect et à la reconnaissance des mouvements, cette fonctionnalité sera présente dans **SportBot**.

Chapter 2. Scénarios d'usage (PAN1)

2.1. Scénarios d'usage

2.2. SportBot

John est jeune, en fin de vingtaine, et n'a jamais été un grand sportif. Il est assez casanier, pas toujours à l'aise avec son corps, et donc pas vraiment enclin à souscrire à un abonnement dans une salle de sport. Il a déjà essayé de faire de l'exercice chez lui, mais en plus de ne pas vraiment savoir quoi faire, la motivation est rapidement venue à manquer. Quand il a entendu parler de SportBot, il s'est dit que c'était le moment ou jamais de se mettre au sport.

John rentre chez lui avec sa nouvelle acquisition, le kit « SportBot ». Il l'ouvre, télécharge l'application associée sur son smartphone, accepte les autorisations pour que SportBot puisse accéder à ses informations sur Facebook, sa musique, et ses informations Youtube, et c'est parti pour une première utilisation. Il met en place la Kinect, pose son téléphone sur le support dédié, puis lance l'application.

SportBot se présente à John : « Bonjour, mon nom est SportBot, je serai ton assistant sportif, mais pas seulement : je serai également ton ami dans tes exercices. Et toi, comment t'appelles-tu ? ». Une interface graphique permet alors à John d'entrer son prénom sur l'application. L'échange continue avec quelques autres questions, notamment en ce qui concerne le poids, la taille, les objectifs sportifs, et les disponibilités de John pour ses exercices. Ce dernier indique à SportBot qu'il désire seulement perdre 1 ou 2 kilos, se maintenir en forme, et qu'il est en général disponible dans l'après-midi. SportBot répond : « Enchanté John, je te propose de commencer tes exercices dès maintenant. De combien de minutes dispose-tu ? ». John entre 30. SportBot répond : « Bien, je vais t'élaborer un programme dans ce temps-là, mais avant j'ai une question pour toi ». Il tire alors au hasard une question dans une liste de questions prédefinies pour apprendre des informations sur son utilisateur. « Quel groupe de musique écoutes-tu ces derniers temps ? ». John entre sur le clavier de son smartphone « Rammstein », puis SportBot enregistre cette réponse dans sa base de données pour pouvoir communiquer par la suite avec John.

La séance peut commencer. SportBot élabore une séance d'exercices pour John, tiré d'une base de 10 à 20 exercices préalablement créée à l'aide d'une API permettant d'apprendre des exercices à SportBot. SportBot montre à John comment faire ses exercices et le suit tout au long de ces derniers, analyse les informations obtenues avec la Kinect, trace un squelette théorique de John, compare son mouvement avec le mouvement à faire en terme de rythme et de forme, et corrige éventuellement John dans ses mouvements. Lors d'une séance de flexions pendant laquelle John était trop courbé en avant, SportBot l'a vite repris : « Ton dos n'est pas assez droit, John ! », avant de le féliciter d'avoir corrigé le tir. Après quelques exercices, il lui demande : « John, je sais que tu aimes bien écouter Rammstein ces derniers temps, ça te dirait que je t'en passe un morceau ? Ou voudrais-tu que je te passe un morceau de Led Zeppelin que tu as aimé sur Facebook ? » John appuie sur le bouton « Rammstein », SportBot cherche alors la musique dans l'application musicale du téléphone et lance un morceau.

Les exercices se poursuivent et John arrive finalement au bout avant les 30 minutes imparties. SportBot lui demande : « Quel exercice as-tu préféré ? ». John sélectionne alors « flexions » parmi la

liste des exercices effectués. « Et quel exercice as-tu trouvé le plus difficile ? ». De la même façon, John sélectionne « pompes ». SportBot lui dit alors « Merci de tes réponses John, c'était une bonne séance aujourd'hui, on se voit à ta prochaine séance », puis l'application se déconnecte.

Deux jours plus tard, dans l'après-midi, John a 20 minutes pour faire du sport, et cela tombe bien car SportBot vient de lui envoyer une notification sur son smartphone pour lui proposer de faire une séance d'exercices. Ni une ni deux, il pose son téléphone sur le support après avoir mis en place la Kinect et lance l'application. « Bonjour John, es-tu prêt pour une nouvelle séance d'exercice ? Combien de minutes as-tu devant toi ? ». John entre 20. « Bien John, je vais te faire un planning d'exercices dans le temps imparti en me servant des indications que tu as déjà pu me fournir, d'ailleurs j'ai une question pour toi ». SportBot va alors tirer au hasard une autre question dans la liste : « Est-ce-que tu regardes des vidéos sur Youtube ? ». Suite à la réponse affirmative de John, il poursuit : « Quel youtuber préfères-tu regarder en ce moment ? ». John entre « MrAntoineDaniel ». « C'est un choix intéressant John, nous pouvons commencer les exercices maintenant si tu veux ». Les exercices s'enchaînent, SportBot lui demande ensuite : « John, souhaiterais-tu écouter un peu de musique ou regarder une vidéo sur Youtube pendant que nous continuons l'entraînement ? », John sélectionne « musique ». « Bien, tu m'as dit il y a deux jours que tu aimes bien Rammstein, je vais t'en lancer un morceau ». Les exercices se poursuivent et John en arrive à bout, puis SportBot s'éteint après les deux questions finales.

Trois jours plus tard, John fait une sortie dans un parc et lance son application SportBot. « Bonjour John, tu n'es visiblement pas chez toi, ou ta Kinect n'est pas allumée. Désires-tu commencer des exercices maintenant ? ». John appuie sur « Oui ». « Bien, combien de minutes as-tu devant toi ? ». John entre 30. « Bien, je vais élaborer une série d'exercices que tu peux faire en extérieur, cependant sache que je ne pourrai pas t'assister dans tes mouvements ici ». John enchaîne ensuite les exercices, guidé par l'application sur son smartphone.

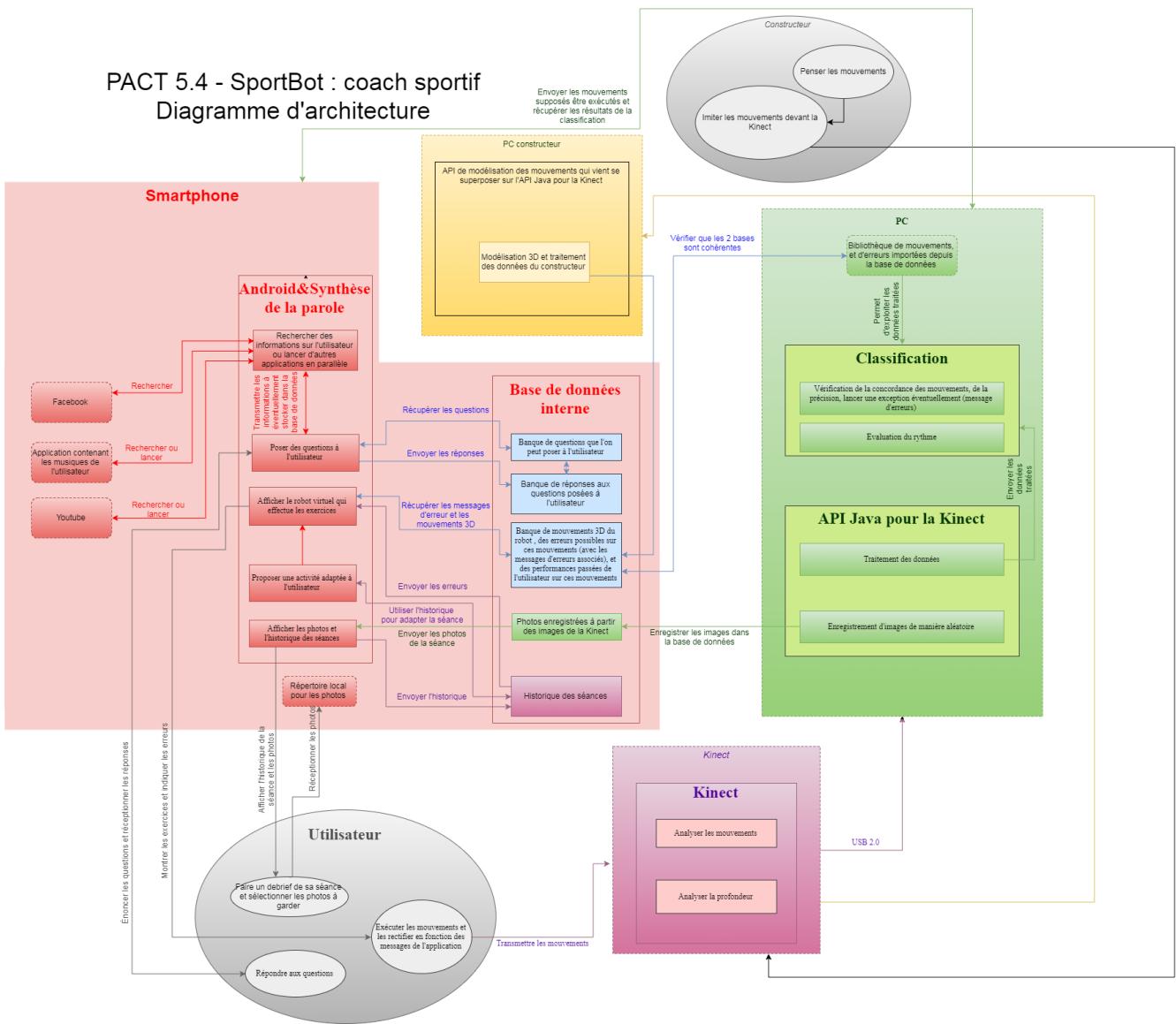
A la séance suivante, il décide de passer à des exercices de plus haut niveau que lui propose l'application qui ressent les progrès du jeune homme. SportBot lui propose d'adopter des positions statiques plus techniques durant ses exercices. John se rappelle alors que SportBot possède une fonctionnalité pour prendre des photos des séances d'exercices de façon régulière via la Kinect. Celle-ci prend des photos de façon à rendre compte d'un maximum de détails de la session d'exercices. SportBot montre, à l'issue de la séance, dans l'application, les différentes photos prises. Au premier coup d'oeil, John semble ravi, les photos le montrent en train d'adopter des poses assez spectaculaires qu'il voudrait bien partager pour vanter sa progression époustouflante, sur les réseaux sociaux notamment. Or, parmi les photos prises, il y en a de bien moins flatteuses par rapport à ses attentes. Même s'il peut choisir les photos à garder à la fin de la séance, il émet un début d'agacement, pensant que l'application pourrait lui permettre de minimiser tous les efforts extra-sportifs à fournir.

John imagine alors qu'au lieu de taper sur son appareil portable pour sélectionner ses photos, il puisse demander à voix haute à SportBot de prendre une photo instantanément. En particulier lorsqu'il arrive à se tenir dans une position extrême dans un court laps de temps, il voudrait que SportBot capture ce moment unique. De plus, il n'aurait pas à s'embêter en fin de séance.

Chapter 3. Architecture du projet (PAN1)

3.1. Schéma d'architecture

3.1.1. Description des blocs



Smartphone :

- Va contenir une application SportBot qui va permettre à l'utilisateur de voir les exercices qu'il doit faire, montré par un robot virtuel, qui lui montrera également ses erreurs, on pourra également l'utiliser pour interroger l'utilisateur, elle affichera également des photos et l'historique de la séance à l'utilisateur à chaque fin de séance. On pourra également l'utiliser pour sauvegarder des photos sur le téléphone en fin de séance. L'application cherchera également des informations sur l'utilisateur en ligne, ou sur son téléphone, pour pouvoir communiquer avec lui (en utilisant par exemple ses musiques favorites sur Facebook).

Utilisateur :

- Il effectue les mouvements que lui montre l'application et interagit avec cette dernière en répondant à ses questions (en entrant des chaînes de caractères ou en cliquant sur des boutons), ou choisit où enregistrer des images en fin de séance.

Kinect :

- Elle analyse les mouvements de l'utilisateur et la profondeur en envoyant un flux de données composé d'images, de profondeur, et un squelette via une caméra couleur RGB et un capteur de profondeur (le son ne nous intéresse pas dans le cadre de notre projet) .

PC Utilisateur :

- Il traite les informations arrivant depuis la Kinect via une API Java et les envoie au module de classification qui se charge de déterminer si le mouvement est bien exécuté ou non, et détermine l'erreur qui existe s'il y en a une à partir des informations des mouvements d'une bibliothèque locale qui correspond à l'exportation de la base de données des mouvements. L'API Java pour la Kinect envoie également régulièrement des images à la base de données qui permettront à l'utilisateur de pouvoir en garder certaines s'il le souhaite en fin de séance.

Constructeur :

- Il conceptualise les mouvements et les erreurs associées et les effectue devant le Kinect.

PC constructeur :

- Une API de modélisation des mouvements effectués par le constructeur, en surcouche de l'API Java pour la Kinect permet de transformer les mouvements de l'utilisateur en fichier de mouvement et erreurs associées que l'on peut ensuite stocker sur la base de données.

Base de données :

- Elle va permettre le stockage des toutes les données mise en jeu, que ce soit les questions que l'on peut poser à l'utilisateur, les réponses (et les informations éventuellement récupérer en ligne (sur Facebook) ou sur son smartphone), les mouvements 3D, les photos enregistrées par la Kinect et l'historique des séances dans un format décrit dans les interfaces.

Interfaces entre Smartphone et Base de données :

- Communication Android avec la Base de données stockée dans un fichier texte sur le Smartphone
- Les photos seront stockées au format JPG dans une table qui leur sera consacrée
- Chaque historique d'une séance sera stockée sous un format JSON dans une table qui leur sera consacrée
- Les questions seront stockées dans une table qui leur sera consacrée (index, nom, type (entier désignant si la question est une question de première utilisation, de début de séance, de fin de séance, ou à poser pendant la séance (dans ce cas un code de type « \numéro d'index correspondant à une partie de réponse » peut être utilisé dans le « nom » qui sera ensuite

complété par des éléments d'une réponse (ou d'éléments obtenus sur Facebook, etc. de l'utilisateur) pour former une question (on utilisera un Regex sur le smartphone avec Java)))

- Les réponses seront stockées dans une table qui leur sera consacrée (index, réponse (texte, tableau JSON qui contient les différents éléments de la réponse, fichier mp3), entier qui détermine sous quel format la réponse est stockée, index de la question correspondante, réponse de l'utilisateur (booléen valant True si c'est le cas, False si c'est une information ici de Facebook par exemple))
- Les mouvements 3D seront stockés sous forme d'un fichier texte par mouvement contenant le mouvement bien réalisé, les mouvements avec erreur (avec les messages d'erreur associés et les highlights que feront une caméra sur le robot 3D pour montrer cette erreur), ainsi qu'un code entier pour désigner chaque possibilité de mouvement. La table correspondante sera de la forme (index, fichier texte, codage MD5 du contenu du fichier txt)

Interface interne au PC utilisateur :

L'API Java pour la Kinect envoie des données traitées au module de classification qui correspond au format sous lesquels les mouvements sont enregistrés pour effectuer la comparaison, le module de classification aura accès aux données et aura l'information du mouvement à traité par le Smartphone (communication client serveur en Wi-Fi)

Interface entre Kinect et PC utilisateur :

USB 2.0 (la Kinect envoie des données sous formes que l'on peut récupérer sous forme de tableau sous Java avec la librairie J4K, à chaque frame traité on reçoit une image couleur, une image de profondeur, et des données de position de points clés des squelettes éventuellement détectés (6 au maximum) ainsi que des rotations entre ces points, nous utiliserons seulement l'image couleur et les squelettes)

Interface entre Kinect et PC constructeur :

USB 2.0 (la Kinect envoie des données sous formes que l'on peut récupérer sous forme de tableau sous Java avec la librairie J4K, à chaque frame traité on reçoit une image couleur, une image de profondeur, et des données de position de points clés des squelettes éventuellement détectés (6 au maximum) ainsi que des rotations entre ces points, nous utiliserons seulement l'image couleur et les squelettes)

Interface entre Utilisateur et Kinect :

Mouvements de l'utilisateur bien placé devant la Kinect et seul (pour détecter un squelette, et pour que ce squelette soit unique)

Interface entre Constructeur et Kinect :

Mouvements de l'utilisateur bien placé devant la Kinect et seul (pour détecter un squelette, et pour que ce squelette soit unique)

Interface entre Smartphone et Utilisateur :

Écran tactile du smartphone qui permet à l'utilisateur d'entrer des réponses à des questions, de

sélectionner différentes options, de voir les exercices effectués par le robot virtuel, de voir les erreurs qu'il a faites, et de voir l'historique de séance

Interface entre Smartphone et PC utilisateur :

- Transmet via du Wi-Fi (Communication Client Serveur) les mouvements supposés être exécutés sous forme de nombre entier (index du mouvement dans la base de données), avec un entier correspondant au nombre de fois où il doit être exécuté (Smartphone vers le PC utilisateur)
 - Transmet via du Wi-Fi (Communication Client Serveur) les codes issus de la classification (index du mouvement de la liste des possibilités (parmi erreurs et mouvement bien exécuté) correspondant au mouvement effectivement effectué par l'utilisateur) avec l'index du mouvement correspondant

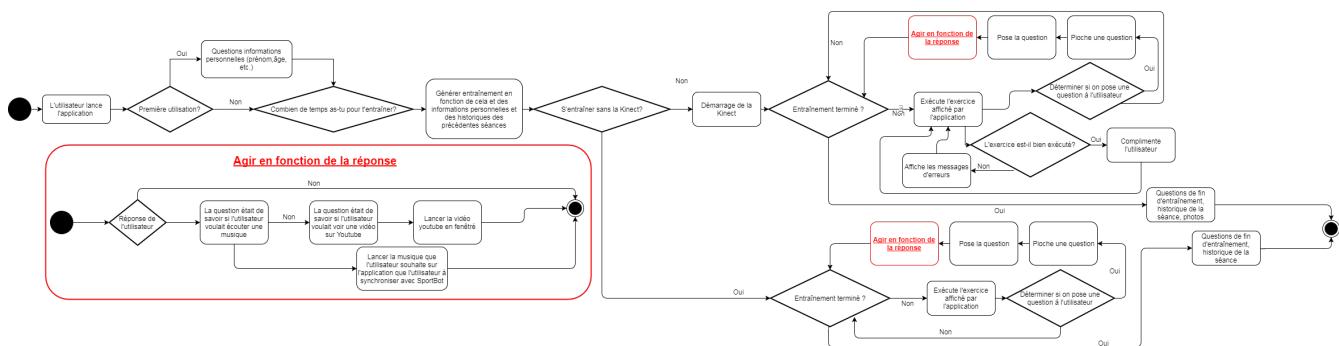
Interfaces entre PC utilisateur et Base de données(interne au Smartphone) :

- Communication Java à l'aide du Wi-Fi en communication client-serveur entre le PC utilisateur et le Smartphone pour atteindre la base de données
 - Dans un dossier du PC utilisateur on stocke localement la table des mouvements de la base de données, on vérifie la cohérence à l'aide des codages MD5 des fichiers avec les codages MD5 correspondant sur la base de données, s'ils ne sont pas identiques on recharge le mouvement, on télécharge les nouveaux mouvements et supprime ceux qui ont éventuellement été supprimés
 - On enregistre des images du flux de la Kinect au format JPG

Interfaces entre PC constructeur et Base de données(interne au Smartphone)

- Communication Java à l'aide du Wi-Fi en communication client-serveur entre le PC constructeur et le Smartphone pour atteindre la base de données
 - Les mouvements 3D seront stockés sous forme d'un fichier texte par mouvement contenant le mouvement bien réalisé, les mouvements avec erreur (avec les messages d'erreur associés et les highlights que feront une caméra sur le robot 3D pour montrer cette erreur), ainsi qu'un code entier pour désigner chaque possibilité de mouvement. La table correspondante sera de la forme (index, fichier texte, codage MD5 du contenu du fichier txt)

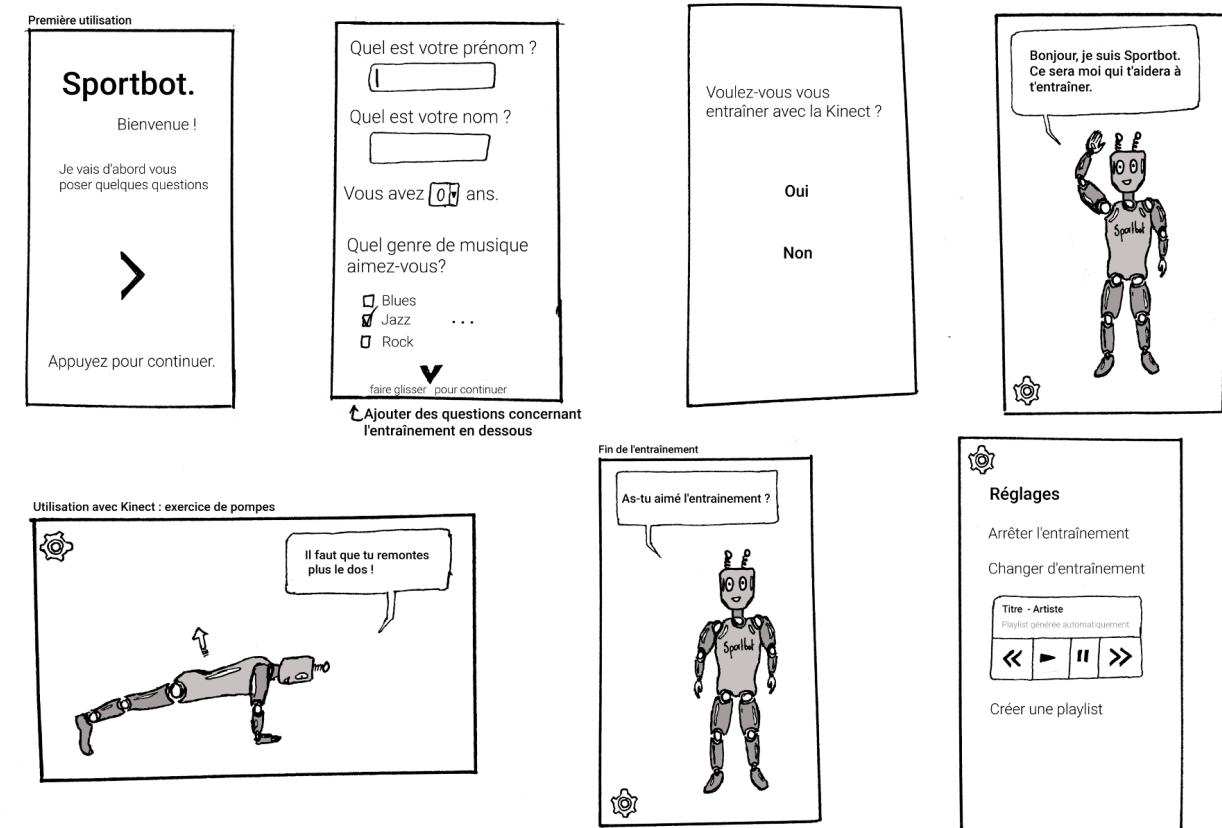
3.2. Diagramme de séquence



3.3. Interface utilisateur graphique

Voici une story board illustrant l'IHM android avec laquelle l'utilisateur pourra interagir. Elle montre également ce qu'il se passe lors de la première utilisation de l'application.

Storyboard SportBot



3.4. Tableau détaillé des tâches

Tâche	Description	Intégré au PAN3	Groupe
T1	Base de données (interne au module Android)		Florian
T1.1	Création de la structure (tables)	x	
T1.2	Définir le format de stockage des mouvements	x	
T1.3	Mise en place de classes Java permettant de récupérer les données via Android	x	

Tâche	Description	Intégré au PAN3	Groupe
T1.4	Création des classes Java permettant de récupérer les données de la base	x	
T1.5	Élaboration liste des questions et classification		
T2	Application Android		Anas/Antonin
T2.1	Conception interface graphique	x	
T2.2	Mise en place interface graphique	x	
T2.3	Robot en mouvement		
T2.4	API synthèse vocale		
T3	Synthèse 3D		Anas/Ali
T3.1	Modèle permettant d'enregistrer mouvements 3D	x	
T3.2	Implantation du modèle d'enregistrement sous Java à partir de l'API Java pour la Kinect	x	
T3.3	Mise en place de caméras virtuelles permettant de montrer à l'utilisateur des highlights de ses erreurs		
T4	Kinect		Axel/Christophe
T4.1	Suivi de squelette dans différentes conditions d'éclairage et tests		
T4.2	Détection des cas d'erreurs (sortie de cadre...)	x	
T4.3	Reconnaissance des mouvements	x	
T4.4	Mesure des performances	x	
T5	Communication client-serveur		Antonin/Axel

Tâche	Description	Intégré au PAN3	Groupe
T5.1	Descriptions de toutes les commandes et réponses possibles entre clients et serveurs	x	
T5.2	Apprendre à mettre en place une communication Wi-Fi entre 2 interfaces	x	
T5.3	Mettre en place la transmission des informations définies dans les interfaces entre le PC utilisateur et le Smartphone de l'utilisateur		
T6	Récupération infos utilisateur		Florian
T6.1	Récupérer intérêts Facebook		
T6.2	Récupération préférences musiques sur le smartphone		
T6.3	Récupération abonnements/historique YouTube		
T7	Entretiens semi-directifs	x	Axel/Christophe
T7.1	Contacter cibles entretiens et planifier entretiens	x	
T7.2	Réaliser entretiens	x	
T7.3	Rédiger synthèse entretiens	x	
T8	Classification	x	Florian/Christophe
T8.1	Identifier la distance à prendre pour comparer	x	
T8.2	Rechercher des séquences exploitables pour faire nos tests	x	
T8.3	Implanter le pseudo code classification en exploitant les outils de la programmation dynamique	x	

Tâche	Description	Intégré au PAN3	Groupe
T8.4	Réaliser des tests avec la distance choisie	x	

Chapter 4. Organisation du projet (PAN1)

4.1. Diagramme de planification temporel des tâches

L'ensemble du déroulement de PACT est disponible sur le [site pédagogique](#).

Toutes les tâches et sous-tâches du projet doivent apparaître dans un diagramme où figurent en abscisses les dates des 4 PANs et en ordonnées les tâches et sous-tâches numérotées. Remplir en couleur les cases où la tâche est active en prenant garde aux dépendances entre tâches. Indiquer via des flèches le moment auquel la tâche entre dans la phase « intégration » ou quand deux tâches indépendantes doivent se synchroniser.

A chaque PAN, sera vérifié l'avancement des tâches selon le schéma prévisionnel. Celui-ci peut être mis à jour, avec l'accord des experts concernés, en cas d'imprévu.

Note 1 : cf. (pour un exemple, mais il ne vous est pas demandé d'utiliser un logiciel pour réaliser votre diagramme, un simple tableur suffit).

Note 2: Vous avez des références d'outils que vous pouvez utiliser dans la liste des composants méthodologiques.

Note 3 : ne pas oublier d'inclure les tâches nécessaires à l'étude bibliographique, à la préparation des livrables (rapports, présentations, vidéo) et à l'intégration des modules dans le prototype.

Tâche	PAN1 (27/11/17)	PAN2 (29/01/18)	PAN3 (19/03/18)	PAN4 (14/05/18)
T1				
T1.1			⇒	
T1.2			⇒	
T1.3			⇒	
T1.4			⇒	
T1.5				
T2				
T2.1			⇒	
T2.2			⇒	
T2.3			⇒	
T2.4				⇒
T3				
T3.1			⇒	
T3.2			⇒	
T3.3				⇒
T4				
T4.1			⇒	

Tâche	PAN1 (27/11/17)	PAN2 (29/01/18)	PAN3 (19/03/18)	PAN4 (14/05/18)
T4.2			⇒	
T4.3			⇒	
T4.4				
T5				
T5.1				
T5.2				
T5.3			⇒	
T6		=⇒ T2		
T6.1			⇒	
T6.2			⇒	
T6.3			⇒	
T7				
T7.1				
T7.2				
T7.3				
T8				
T8.1				
T8.2				
T8.3			⇒	
T8.4			⇒	

Légende :

- => la tâche entre dans la phase intégration
- ==> Tx la tâche entre en synchronisation avec la tâche Tx

4.2. Répartition des élèves par module

Nom	Communication client serveur	Android	Classification	Synthèse 3D	KINECT	Module SES	Test & Intégration
Nom Expert	DUFOUR Jean-Claude	DUFOUR Jean-Claude	ROUX Michel	THIERY Jean-Marc	LE FEUVRE Jean	DIMINESCU Dana	DUFOUR J.C.
BONNARD Florian			X				X
TORCHY Axel	X				X	X	

Nom	Communication client serveur	Android	Classification	Synthèse 3D	KINECT	Module SES	Test & Intégration
BENJELLOUN Ali				X			X
GODARD Antonin	X	X					
VUONG Christophe			X		X	X	
BOUZAFOUR Anas		X		X			

4.3. Plans de test (PAN2+)

Plan tests proto allégé (dans l'ordre chronologique) : (voir en annexe Intégration et tests pour les détails)

Rappel : Les conclusions du rapport Kinect ont été prises en compte, à savoir une distance préconisée entre 1m75 et 3m50. On restera aux alentours de 2m pour nos divers tests.

Intitulé du test	Configuration du test	situation/contexte	action ou entrée à appliquer	réaction ou sortie attendue
1 : Kinect allégée + Synthèse 3D allégée	Kinect pour lire depuis un PC ou sauvegarder un squelette	Kinect en marche, visualisation sur le PC via un programme Java des mouvements détectés et enregistrement délimité en temps	Flux Kinect associé à un mouvement simple durant le temps impari	2 tableaux stockés dans un fichier txt: une ligne par trame contenant [positions[0],x],positions[1].y,...,positions[20].z] pour les 20 points d'intérêts et un tableau indiquant la fiabilité des coordonnées

Intitulé du test	Configuration du test	situation/contexte	action ou entrée à appliquer	réaction ou sortie attendue
2 : Kinect allégée + classification	Transmettre les données de squelette pour réaliser le traitement de classification	Méthode de classification utilisant le Dynamic Type Warping au point, soft de Synthèse 3D à disposition, des fichiers référence pour pouvoir comparer avec l'utilisateur	Un fichier texte(utilisateur) contenant tableau de 60 flottants, pour les 20 positions, 3 axes [visible[0],...,visible[20]] //visible[i] vaut 1 si la position est calculée par la Kinect, 2 si elle est visible par la Kinect (⇒ plus fiable)	Un fichier texte contenant la valeur de l'erreur finale et les valeurs DTW des différents points d'intérêts
3 : Classification + Synthèse 3D allégée	Rassembler du côté constructeur les données d'un mouvement référence et des erreurs associées dans un fichier de mouvements	La classification a été modifiée selon cette nouvelle structure, on dispose de fichiers texte de séances d'exercice que l'on va utiliser comme des références	Des fichiers texte du mouvement et 2 à 3 erreurs associées	un fichier mouvements sous format txt regroupant toutes les données évoquées précédemment avec un niveau d'appréciation par rapport à la réussite du mouvements
4 : Synthèse 3D allégée	Découper les enregistrements de mouvements en plusieurs morceaux	Des fichiers texte pour lesquels le nombre de trames est variable	Un fichier texte d'un mouvement répété plusieurs fois	Un fichier texte qui ne contient qu'une répétition du mouvement
5 : Kinect + Classification + Synthèse 3D	Enregistrer plusieurs classes de mouvements du côté PC constructeur	Tous les tests précédents ont fonctionnés	Un mouvement réalisé par l'utilisateur	Le résultat de la classification sous forme d'index d'un élément de la classe dudit mouvement qui correspond à son plus proche voisin à chaque répétition du mouvement

Intitulé du test	Configuration du test	situation/contexte	action ou entrée à appliquer	réaction ou sortie attendue
6 : Communication Client-Serveur + Android + PC(Test 1)	Transmettre des données traitées par le PC	Tous les tests précédents sont réussis, les fichiers de classes de mouvements référence sont dans le smartphone, et les softs sont sur le PC	Lancer des messages de notifications via l'interface Android	Vérifier que les données à communiquer entre le PC et le smartphone sont arrivées en l'état via des messages de suivi sur le PC et sur le smartphone
7 : Communication Client-Serveur + Android + PC(Test 2)	Vérifier la synchronisation entre le PC et le smartphone au niveau de la base de données	La transmission entre les 2 doit être fonctionnelle	3-4 mouvements côté smartphone à envoyer au PC	Une base de données qui se crée côté PC
8 : Communication Client-Serveur + Android + Kinect allégée	Vérifier que l'application peut lancer la Kinect	La Kinect doit être sous tension et l'ordinateur doit être allumé pour qu'il y ait communication	L'utilisateur enclenche le mode avec Kinect	La Kinect se met en marche(infrarouge), et on détecte un squelette via le viewer 3D
9 : Communication Client-Serveur + Android + Kinect allégée + classification	Vérifier que le traitement de la classification arrive bien au smartphone	L'application est en cours d'utilisation, la Kinect est en marche, l'utilisateur fait ses mouvements	Dess mouvements utilisateurs en continu copiant le bot du viewer 3D	Des messages de suivi de classification en continu et en toute fluidité

Plan tests proto final (dans l'ordre chronologique) : (voir en annexe Intégration et tests pour les détails)

Intitulé du test	Configuration du test	situation/contexte	action ou entrée à appliquer	réaction ou sortie attendue
10 : Android	Vérifier que l'application a bien accès aux musiques stockées sur le smartphone pour lecture	L'application est en pleine utilisation	Requête de lecture de musique	La musique qui se lance puis s'arrête et on revient à la fenêtre principale de l'application
11 : Android	Vérifier l'accès aux pages likées sur Facebook de l'utilisateur	Lorsque l'application se lance initialement	Requête pour avoir accès à ces informations	Affichage d'un message avec le nom de la dernière page likée par exemple

Intitulé du test	Configuration du test	situation/contexte	action ou entrée à appliquer	réaction ou sortie attendue
12 : Android	Lancement de la synthèse vocale pendant les transitions	Lorsque l'application passe d'un exercice à l'autre, on voit toujours le viewer	Le premier exercice s'est arrêté, la classification est en pause	La synthèse vocale doit se déclencher et lire un message prédéfini

4.4. Diagramme d'avancement des tâches (PAN2+)

PAN2

Module	Avancement Prévu (%)	Avancement Réel (%)	Temps passé (h)	Description brève du travail effectué, analyse des écarts constatés
Classification	80%	70%	6	Les tâches T8.1, T8.2, T8.3 sont terminées – La tâche T8.4 est en cours – Un rapport a été transmis à l'expert – 2 réunions ont eu lieu pour tester la distance choisie, discuter de l'avancement et se mettre d'accord sur le code. On a donc pu faire suffisamment de tests, mais des paramètres inopportunus nous ont poussé à déterminer un cadre plus précis pour l'utilisation de la Kinect dans notre projet.
Kinect	30%	50%	8	Les tâches T4.2, T8.3, T8.3 sont terminées – Les tâches T4.1 et T4.4 sont en cours – 1 réunion a eu lieu pour tester le code, une autre a été organisée pour parler du robot 3D - Présentation du soft Kinect avec un robot 3D primitif
Entretiens semi-directifs	100%	100%	10	Toutes les tâches sont terminées – Un rapport comprenant des entretiens oraux a été transmis aux experts – 1 soutenance ont eu lieu pour discuter de l'exploitation des résultats obtenus à l'issue des entretiens.
Android	50%	40%	8	Création du squelette de l'application : menu de première ouverture, entrée des infos utilisateurs, menu de choix kinect (oui/non), menu principal, menu de réglages. Tâches T2.1.
Communication client-serveur	80%	80%	8	Mise en place de la communication client/serveur via wifi. Réception et envoi de messages. Tâche T5.1, T5.2, T5.3.
Synthèse 3D	80%	70%	10	T3.1, T3.2

PAN3

Modul e	Avance ment Prévu (%)	Avance ment Réel (%)	Tem ps passé (h)	Description brève du travail effectué, analyse des écarts constatés
Classification	100%	100%	10	Toutes les tâches sont terminées – Un rapport a été transmis à l'expert – 1 réunion
Kinect	90%	80%	12	Les tâches T4.2, T8.3, T8.3 sont terminées – La tâche T4.1 est en cours – 1 réunion a eu lieu pour tester le code, une autre a été organisée pour parler du robot 3D - Présentation du soft Kinect avec un robot 3D primitif
Andro id	80%	70%	8	Intégration de la 3D dans le menu principal. Mise en place de création de dossier dans le stockage externe (téléphone). Tâches T1.2, T2.2
Comm unicati on client-serveu r	80%	80%	4	Intégration de la communication
Synthè se 3D	80%	70%	Idem que pan 2 + intégration android	

PAN4

Modul e	Avance ment Prévu (%)	Avance ment Réel (%)	Tem ps passé (h)	Description brève du travail effectué, analyse des écarts constatés
Classification	100%	100%	1	Toutes les tâches sont terminées – Un rapport a été transmis à l'expert – 1 réunion
Kinect	100%	100%	5	Toutes les tâches ont été réalisées. Les exercices finaux ont été édités
Andro id	100%	95%	8	Toutes les tâches ont été réalisées. Les fonctions de likes Facebook restent à être implémentées
Comm unicati on client-serveu r	100%	100%	1	La communication client-serveur est au point

Modul e	Avance ment Prévu (%)	Avance ment Réel (%)	Tem ps passé (h)	Description brève du travail effectué, analyse des écarts constatés
Synthèse 3D	100%	100%	3	Intégration de highlights pour la notification d'erreurs de l'utilisateur

Bibliographie (PAN1+)

Note : Liste des références exploitées. Une référence complète donne titre, auteur(s), date, journal, revue, source de publication, titre de conférence, numéro, pages. Une webographie est aussi envisageable : titre, auteur, date, page web

- Design Guideline : **Android**, <https://developer.android.com/design/index.html>
 - J4K Java Library : **University of Florida, Digital Worlds Institute, USA**
<http://research.dwi.ufl.edu/ufdw/j4k/index.php>
 - Analyse de séquences vidéo : **Christian Wolf**, 2014, <http://liris.cnrs.fr/christian.wol>
 - Dynamic Programming : **Avrim Blum**, 2009, <http://www.cs.cmu.edu/~avrim/451f09/lectures/lect1001.pdf>
 - Create Avatars using Kinect in Real-time: **Angelos Barmpoutis**, 2012, article de l'IEEE, 10 pages
 - JDBC : **Java DataBase Connectivity**, <http://www.oracle.com/technetwork/java/overview-141217.html>
-

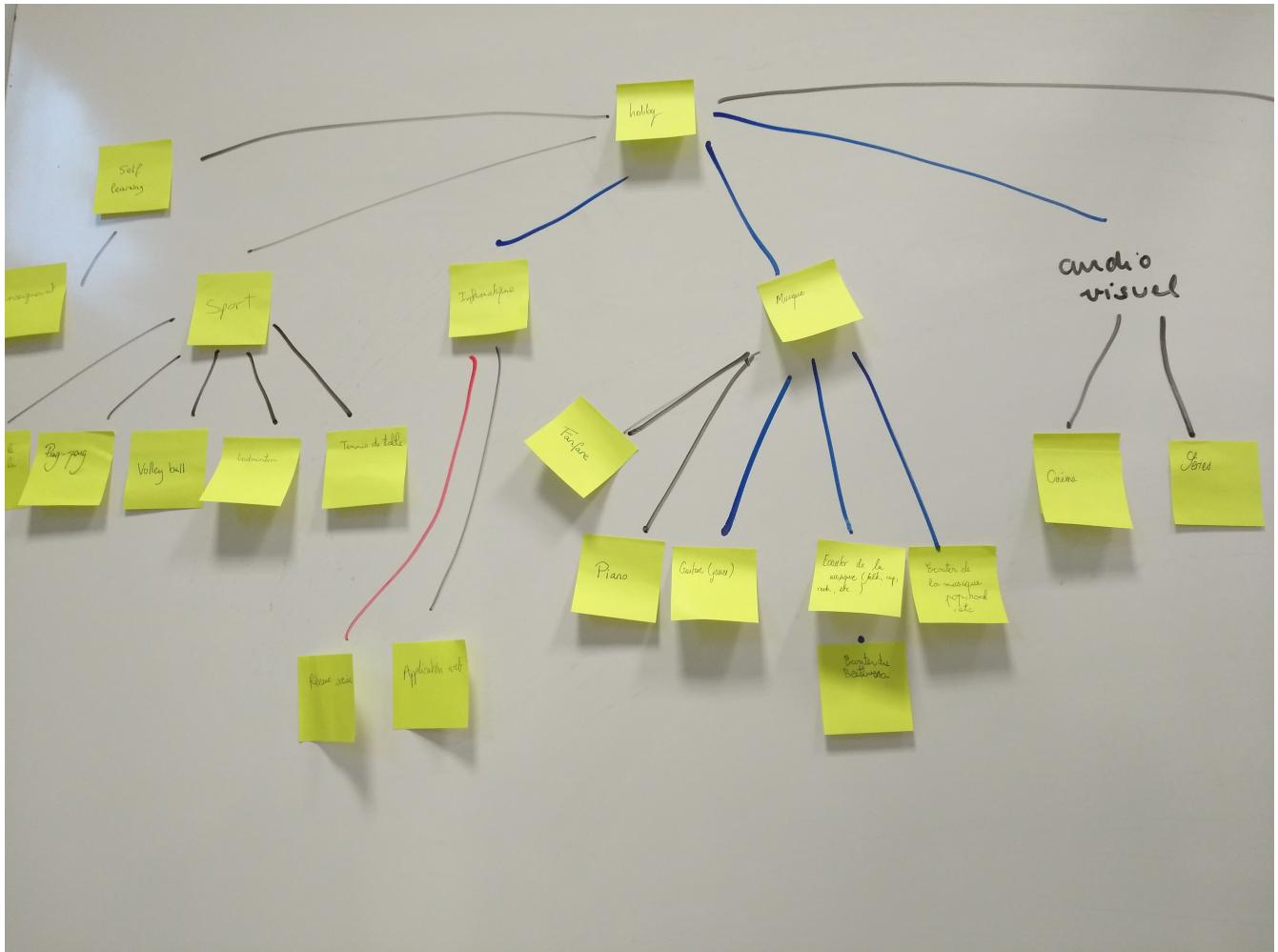
Chapter 5. Annexes

5.1. Fiche d'identité du groupe (PAN1)

Le travail en groupe est totalement nouveau pour chacun d'entre nous. Nous avons mis du temps avant de trouver nos marques. Nous nous concertons assez régulièrement, notamment en utilisant les moyens que nous offrent les réseaux sociaux. Nous avons pourtant tendance à travailler en sous-groupes, mais nous tâchons de diviser intelligemment le travail pour la rédaction du scénario comme pour l'élaboration des différents diagrammes. Dans l'ensemble, nous sommes assez motivés, même si nous n'avons pas tous nécessairement les mêmes connaissances en informatique. Chacun apporte ses idées et on essaie d'en discuter pendant les séances avec notre tuteur par exemple. Pour trouver notre projet, on s'est concerté sur nos goûts respectifs. La musique, le sport et l'enseignement ont été largement cités, ce qui nous a amené à penser l'application SportBot qui regroupe ses thèmes. Nous avons des personnalités assez différentes et cela amène son lot de surprises. Nous ne partageons pas forcément des méthodes de travail communes. Néanmoins, dès que le projet a été bien défini, nous nous sommes mis d'accord tous ensemble sur une organisation efficace du travail à adopter. Nous avons un coordinateur en la personne de Florian qui lance généralement les discussions sur le PACT. Or, à présent chacun a pris conscience de son rôle à jouer dans le projet grâce aux modules. On devient alors tous plus actifs et plus impliqués dans le projet. Naturellement, la communication est devenue au fil du temps plus satisfaisante dans le groupe.

Le seul membre du groupe qui a un peu d'expérience en informatique est Florian, il est très à l'aise dans l'utilisation des outils, ce qui permet aux autres d'en apprendre à chaque réunion. Christophe est doté d'un tempérament sérieux, il n'aime pas perdre de temps dans le travail, ce qui dynamise les conversations et donne souvent le tempo. Ali, Antonin, Axel et Anas sont tout aussi investis. Ils apportent des ondes positives au groupe et des idées qui font avancer le projet.

Cartographie du groupe



5.2. Modifications (PAN2+)

5.2.1. Modifications de fond

Tableau des modifications de fond apportées au projet avec validation des experts et encadrant informatique

libellé / date	Description brève	Validé par :
Retrait du module base de données à la présentation PAN1	La base de données sera stockée localement	Jean-Claude Dufourd

5.2.2. Modifications du rapport

Vous noterez dans cette section les modifications apportées au rapport depuis le PAN précédent. Si votre planification temporelle a été modifiée, vous laisserez l'ancienne planification dans cette annexe.

Modifications du rapport au PAN2

- Arrangement du diagrammes d'architecture par rapport à la modification de fond au PAN1
- Révision du document interfaces.adoc
- Mise à jour du diagramme de tâches

- Ajout de preuves de tests dans le cadre du module classification et du module Kinect
- Ajout d'un rapport de classification expliquant la démarche et les tests réalisés
- Ajout d'une bibliographie Kinect et d'une étude sur les caractéristiques morphologiques susceptibles d'influencer les résultats de notre algorithme de classification utilisant les données de flux Kinect

Modifications du rapport au PAN3

- Mise à jour des rapports du module Kinect et du module classification
- Ajout en avancement(annexe) d'un descriptif du module synthèse 3D
- Mise à jour des comptes-rendus de réunions
- Mise à jour de l'avancement en terme de % dans chaque module

Modifications du rapport au PAN4

Hac habitasse platea dictumst. Cras quis lacus. Vestibulum rhoncus congue lacus. Vivamus euismod, felis quis commodo viverra, dolor elit dictum ante, et mollis eros augue at est. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nulla lectus sem, tristique sed, semper in, hendrerit non, sem. Vivamus.

5.3. Comptes Rendus de réunions

5.3.1. Réunion du 4/12/2017 (durée 2h30)

- Présents:
 - Axel Torchya
 - Christophe Vuong
 - Florian Bonnard
- Sujets abordés
 - **Classification** : confirmer la distance envisagée
 - **Tests Kinect** : faire tourner les démos de J4K
- Actions à entreprendre:
 - Contacter l'expert Jean Lefevre pour la suite du module Kinect
 - Implémenter les méthodes Java pour la classification.

5.3.2. Réunion du 18/12/2017 (durée 2h)

- Présents:
 - Axel Torchya
 - Christophe Vuong
 - Florian Bonnard

- Sujets abordés
 - **Kinect** : penser comment stocker les données du flux Kinect pour pouvoir les exploiter même sans l'appareil.
 - **Tests de la Kinect** : Discussion sur leur conception
 - **Entretiens SES** : qui interroger ? l'interaction que pourra proposer SportBot
- Actions à entreprendre:
 - Développer les classes en Java qui permettront de récupérer un fichier texte contenant les coordonnées des articulations captées par la Kinect.
 - Commencer à contacter des personnes pour les entretiens semi-directifs.

5.3.3. Réunion du 8/01/2018 (durée 2h)

- Présents:
 - Axel Torchy
 - Christophe Vuong
 - Anas Bouzafour
- Sujets abordés
 - **Rappel des objectifs dans les modules, notamment la synthèse 3D**
 - **Tests de la Kinect** : test avec la Kinect pour valider notre code Kinect.
- Actions à entreprendre:
 - Faire le lien entre le module classification, le module Kinect, et communiquer les résultats obtenus au binôme de synthèse 3D.
 - Faire les entretiens semi-directifs sur un spectre de profils assez large en binôme. =====
Réunion du 15/01/2018 (durée 2h)
- Présents:
 - Axel Torchy
 - Christophe Vuong
 - Florian Bonnard
- Sujets abordés
 - **Kinect** : point sur les tests et et les exercices qui seront faisables pour l'instant
 - **Mise à jour de la grille SES** : questions sur l'interaction que pourra proposer SportBot
 - **Intégration** : que doit-on faire dans les modules Kinect et synthèse 3D avant l'intégration ?
- Actions à entreprendre:
 - Faire le lien entre le module classification, le module Kinect, et envoyer les résultats obtenus au binôme de synthèse 3D.
 - Faire les entretiens semi-directifs sur un spectre de profils assez large en binôme.

5.3.4. Réunion du 22/01/2018 (durée 1h30)

- Présents:
 - Axel Torchys
 - Christophe Vuong
 - Florian Bonnard
- Sujets abordés
 - **Kinect** : Problèmes sur l'estimation de l'erreur par la Kinect
 - **Bilan de ce qui a été fait**
- Actions à entreprendre:
 - Contacter les experts assez rapidement pour partager notre avancement.
 - Interroger les personnes contactées dans le cadre du module SES.

5.3.5. Réunion du 29/01/2018 (durée 15 min)

- Présents:
 - Antonin Godard
 - Jean-Claude Dufour
- Sujets abordés
 - **Android**
 - **Bilan de ce qui a été fait**
- Actions à entreprendre:
 - Avancer en Android

5.3.6. Réunion du 29/01/2018 (durée 15 min)

- Présents:
 - Florian Bonnard
 - Ali Benjelloun Zahar
 - Jean-Claude Dufourd
- Sujets abordés
 - **Intégration**
 - **Plans-tests**
- Actions à entreprendre:
 - Mettre en oeuvre ces plan-tests

5.3.7. Réunion du 29/01/2018 (durée 30 min)

- Présents:

- Axel Torchay
- Christophe Vuong
- Jean Lefeuuvre
- Sujets abordés
 - **PAN2**
 - **Présentation du soft Kinect et des critères d'erreur**
- Actions à entreprendre:
 - Ecrire un compte-rendu de tests de performance de la Kinect.
 - Avancer en synthèse 3D

5.3.8. Réunion du 29/01/2018 (durée 20 min)

- Présents:
 - Florian Bonnard
 - Ali Benjelloun Zahar
 - Anas Bouzafour
- Sujets abordés
 - **PAN2**
 - **Intégration :**
- Actions à entreprendre:
 - Faire un petit bilan de ce qui a été fait en intégration et en synthèse 3D à tout le groupe.

5.3.9. Réunion du 31/01/2018 (durée 45 min)

- Présents:
 - Florian Bonnard
 - Christophe Vuong
 - Michel Roux
- Sujets abordés
 - **PAN2 : classification**
- Actions à entreprendre:
 - Mettre en place notre plan en classification.
 - Faire plus de tests pour confirmer notre avancée

5.3.10. Réunion du 5/02/2018 (durée 20 min)

- Présents:
 - Axel Torchay

- Christophe Vuong
- Julien Morel
- Dana Diminescu
- Sujets abordés
 - **Soutenance SES**
- Actions à entreprendre:
 - Intégrer les remarques des experts à notre projet
 - Faire un compte-rendu succinct aux autres membres du groupe

5.3.11. Réunion du 5/02/2018 (durée 1h30)

- Présents:
 - Axel Torchay
 - Antonin Godard
 - Jean-Claude Dufourd
- Sujets abordés
 - **Communication client-serveur**
 - **Tests Kinect** : distance entre la Kinect et l'utilisateur
- Actions à entreprendre:
 - Proposer une version plus adaptée de la communication client-serveur

5.3.12. Réunion du 5/02/2018 (durée 1h30)

- Présents:
 - Axel Torchay
 - Christophe Vuong
 - Florian Bonnard
- Sujets abordés
 - **Mise à jour du soft Kinect**
 - **Bilan** : à tout le groupe de l'avancement général et du retour des experts.
 - **Tests Kinect** : distance entre la Kinect et l'utilisateur
- Actions à entreprendre:
 - Faire un plan de tests plus poussé
 - Compléter le rapport du module Kinect
 - Utiliser les tests réalisés pour faire de la classification.
 - Proposer une première version en Android
 - Compléter le compte-rendu de tests de performance de la Kinect.

5.3.13. Réunion du 26/02/2018 (durée 1h30)

- Présents:
 - Axel Torchya
 - Christophe Vuong
 - Florian Bonnard
 - Antonin Godard
 - Ali Benjelloun Zahar
 - Anas Bouzafour
- Sujets abordés
 - **PAN3** : faire un bilan de tout ce qui a été fait
 - **Communication entre les différents modules** : clarifier des détails sur l'intégration de la synthèse 3D au projet
- Actions à entreprendre:
 - Organiser ce qui a faire sous forme d'issue.
 - Proposer une interface plus complète en Android
 - Faire des statistiques sur les tests Kinect
 - Réaliser l'intégration de la communication client-serveur
 - Finir le code de synthèse 3D, et l'intégrer au projet

5.3.14. Réunion du 05/03/2018 (durée 45 min)

- Présents:
 - Axel Torchya
 - Christophe Vuong
 - Jean Lefevre
- Sujets abordés
 - **PAN3** : parler du rapport de tests mis à jour
 - **Kinect** : retour sur les études des limitations de la Kinect
- Actions à entreprendre:
 - Mettre à jour le rapport du module Kinect
 - Mettre à jour l'issue pour les tâches du PAN3
 - Pousser sur le dépôt git le code de synthèse 3D

5.3.15. Réunion du 08/03/2018 (durée 30 min)

- Présents:
 - Axel Torchya

- Christophe Vuong
- Florian Bonnard
- Sujets abordés
 - **PAN3** : finir l'intégration
- Actions à entreprendre:
 - Pousser sur le dépôt git ce qui a été fait en intégration
 - Régler quelques détails en classification
 - Scénariser la démo pour la PAN3

5.3.16. Réunion du 12/03/2018 (durée 1h)

- Présents:
 - Axel Torchya
 - Christophe Vuong
 - Christophe Prieur
 - Florian Bonnard
 - Antonin Godard
 - Ali Benjelloun Zahar
 - Anas Bouzafour
- Sujets abordés
 - **PAN3** : parler du rapport de tests mis à jour
 - **Démo** : expliquer ce qui va être fait le jour J
 - **Classification** : régler quelques bugs mineurs
- Actions à entreprendre:
 - Faire les tâches qui restent sur l'issue
 - Finir d'intégrer la partie Android, communication client-serveur avec le soft Kinect

5.3.17. Réunion du 16/04/2018

- Axel Torchya
- Christophe Vuong
- Christophe Prieur
- Antonin Godard
- Sujets abordés
- Bilan travail en équipe : efficacité collaborative, issues sur gitLab, équipe soudée ? *Actions à entreprendre:
- Effectuer les tâches référencées dans les issues

5.3.18. Réunion du 19/04/2018

- Axel Torchya
- Christophe Vuong
- Florian Bonnard
 - Sujets abordés
- Intégration des questions-réponses, détails Android *Actions à entreprendre:
- Implémenter les highlights, et le questions-réponses
- Mettre à jour les issues

5.3.19. Réunion du 20/04/2018

- Axel Torchya
- Christophe Vuong
- Florian Bonnard
- Antonin Godard
 - Sujets abordés
- Tournage de la vidéo, conception du poster *Actions à entreprendre:
- Implémenter le lecteur musique, les textures du bot en Android

5.3.20. Réunion du 02/05/2018

- Axel Torchya
- Christophe Vuong
- Antonin Godard
 - Sujets abordés
- Fin du poster, validation de la vidéo, présentation PowerPoint *Actions à entreprendre:
- Implémenter le lecteur musique, les textures du bot en Android
- Compléter les dernières sections du rapport
- Finir les dernières slides du diaporama

5.4. Synthèse 3D

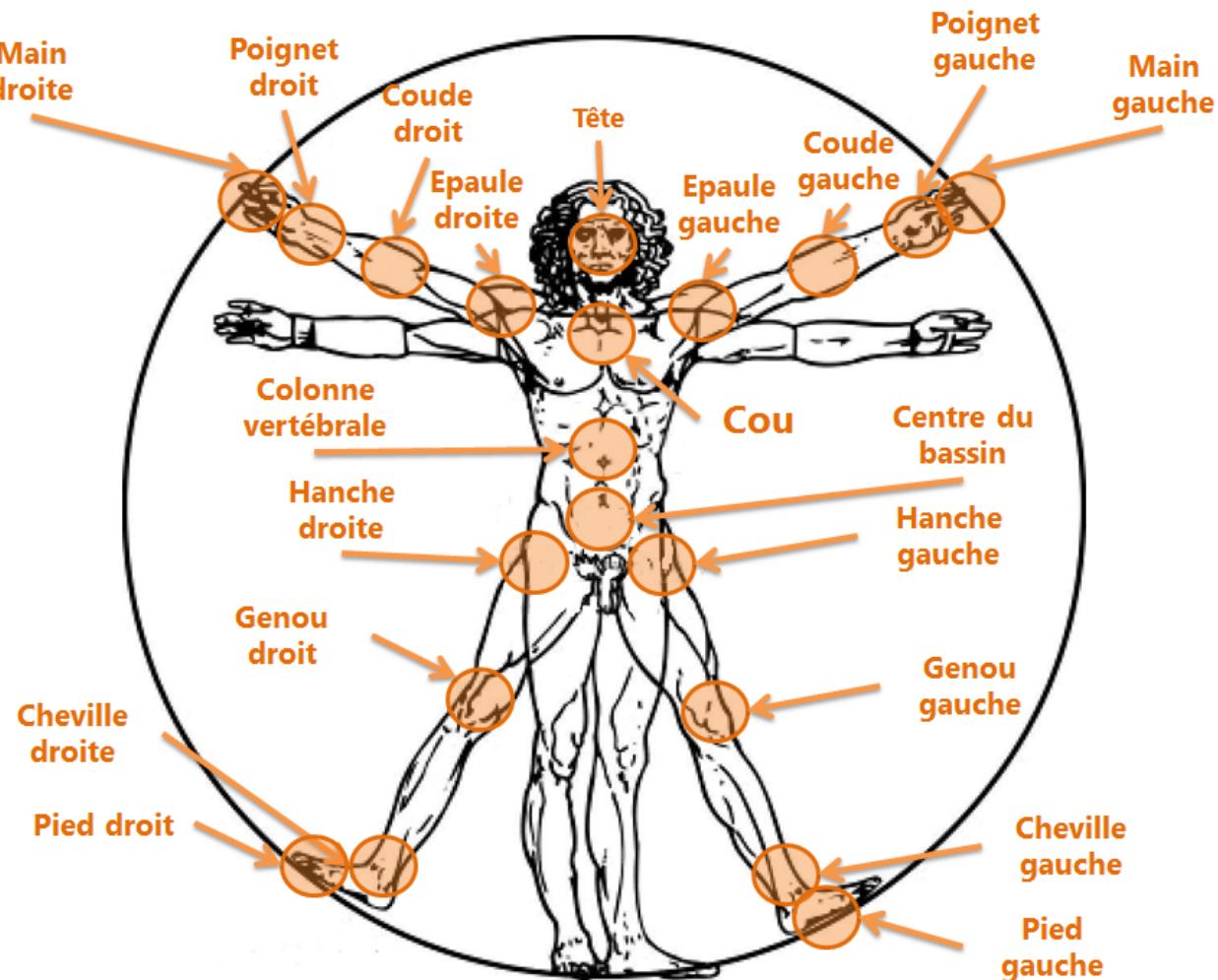
5.4.1. Descriptions

- Réaliser un soft qui permet d'enregistrer les mouvements références(avec erreurs ou sans)
- Concevoir le robot 3D pour l'affichage sur Android

5.4.2. Démarche globale

Récupération du flux Kinect, synthèse de mouvements références et stockage

La conception des mouvements doit être en accord avec les données envoyées en classification. En effet, il s'agit de capter des mouvements via la Kinect. On récupère donc en premier lieu des coordonnées en repère orthonormé(cartésien) pour les 20 points d'intérêts.



La Kinect en envoie 20, mais comme le montre le schéma, les 20 sont suffisants pour faire notre application sportive. On crée donc un fichier mouvement à partir de ces coordonnées. Il est stocké sur les machines sous forme de fichier texte avec la structure suivante : Une suite de séquences (la première étant la séquence de référence (mouvement bien réalisé)) séparée par un espace, de la forme :

- Le nom de la séquence
- Une ou plusieurs lignes de messages se rapportant à la séquence
- 28 lignes d'information pour la classification de la séquence (module classification)
- La ligne "Enregistrement d'un fichier Kinect" qui marque le début de l'enregistrement
- Des coordonnées des 20 membres qui s'enchaînent 2 lignes par 2 lignes (la première pour les 60 coordonnées des points (3*20), la deuxième pour les 20 entiers qui valent 1 ou 2 selon si le point est calculé par la Kinect (donc la position est devinée) ou s'il est réellement vu)

Ce fichier mouvement résultant comporte donc plusieurs séquences de mouvements qui correspond à une même classe d'exercice. Il n'y a qu'une référence juste (la première) et des références erronées (qui ne seront pas prises comme référence tout du moins) en nombre quelconque. Dans le cadre des tests Kinect répertoriés, on s'est assuré d'avoir le même type de mouvement, même si on fait des modèles avec erreur de celui-ci.

Exploitation avec un soft 3D

On a repris 2 modèles de projection utilisés par OpenGL pour tracer des cylindres pour les membres et une sphère pour la tête (perspective et orthogonale). Ces solides permettent de donner la structure du robot affiché en 3D. Pour dessiner les cylindres, on utilise une technique classique qui est d'empiler des triangles. La section circulaire du cylindre est divisée en plusieurs morceaux qui vont être approximés par des triangles. Et les côtés du cylindre sont des rectangles. On fait quelque chose de similaire pour les sphères qui représentent la tête, en utilisant de base la figure de Schläfli {3,4} (cad l'octaèdre régulier), chaque triangle de base est redivisé par récurrence en 4 triangles égaux avec 3 nouveaux sommets qui sont les milieux des arrêtes du triangle original, on effectue ensuite un processus de normalisation en plaçant tous les sommets à distance R du centre de l'octaèdre régulier pour obtenir une sphère.

Il a fallu déterminer la direction de l'axe de révolution du cylindre, en utilisant des angles en coordonnées sphériques:

$$\theta = \arctan\left(\frac{y_{top} - y_{base}}{x_{base} - x_{top}}\right)$$

$$\phi = \arccos\left(\frac{z_{base} - z_{top}}{\sqrt{(x_{base} - x_{top})^2 + (y_{base} - y_{top})^2 + (z_{base} - z_{top})^2}}\right)$$

A partir de ces figures 3D calculées, on a fait une projection du squelette(perspective)[1]. Elle consiste en considérer le point 3D contenu dans une pyramide tronquée qui a dans ce cas 2 bases rectangulaires b_{near} et b_{far} qui vont nous servir de plans de projection. En considérant des relations de similitude, on parvient à exprimer les coordonnées 2D de la projection. Il reste alors à les normaliser, d'où un autre paramètre w introduit. Déterminer ces valeurs caractéristiques de la projection revient à faire passer d'une représentation à l'autre par une matrice de projection avec tous les paramètres cités précédemment[1].

De plus, on a introduit une condition (`faceIsCullable`) pour savoir si un élément (triangle ou rectangle) d'une des formes utilisées (cylindre convexe ou sphère convexe) est face au viewer ou pas (nécessaire de le tracer ou pas) à partir des coordonnées reçues, en utilisant la formule suivante :

$$face_{cul} = (\vec{u} \wedge \vec{v}) \cdot \vec{u}_z$$

avec :

$$\vec{u} = (x_1 - x_0, y_1 - y_0, z_1 - z_0)^T$$

$$\vec{v} = (x_2 - x_0, y_2 - y_0, z_2 - z_0)^T$$

en considérant une série de 3 points d'intérêts de l'élément (le 1er, 2eme et dernier) en question dans le sens trigonométrique.

Si le signe de face_cul est +, le membre est de face et est représenté, sinon il est de dos.

Enfin, on fait une transposition du rendu grâce à la classe Java Windowering et on affiche alors sur le Viewer de la Canvas.

5.4.3. Avancement au PAN3

Toute l'implémentation pour la synthèse 3D a pratiquement été réalisée entre le PAN2 et la PAN3. Il reste des détails à voir sur le design du robot. Il faudrait que celui-ci corresponde à ce qu'on fera en synthèse vocale, et en Android par la suite.

5.4.4. Réalisations au PAN4

- Nous avons implémenté des highlights de la séance avec zoom sur les erreurs de l'utilisateur (manipulation de caméras virtuelles).
- Le squelette de la kinect v2 n'offrant pas des détails sur l'orientation/les angles, l'implementation d'un modèle en.obj ou .max a été délicate.

5.4.5. Bibliographie

[1] http://www.songho.ca/opengl/gl_projectionmatrix.html

5.5. Module Classification

5.5.1. Descriptions

- Déterminer une distance(ou erreur) entre une séquence de mouvement et un mouvement dans la base de données
- Créer des classes de mouvements correspondant à chaque exercice dans la base de données avec dans chaque classe une référence bien exécutée et d'autres séquences de mouvements erronées avec le message d'erreur associée
- Editer les mouvements pour une démonstration et des mouvements tests pour confirmer le code utilisé
- Utiliser les méthodes DTW et les données fournies par le module pour réaliser le soft synthèse 3D

Dans le rapport qui suit, on a d'abord étayé la démarche globale dont les détails ont été motivés par les tests réalisés tout au long du module.

5.5.2. Démarche globale

Dans le cadre du module, nous avons envisagé d'implémenter une méthode utilisant la programmation dynamique adapté à notre programme. En effet, en parallèle, nous avons développé des classes Java pour récupérer le squelette de notre utilisateur, et en particulier les coordonnées 3D des articulations récupérées par la Kinect. On a par ailleurs pu relire un fichier enregistré et affiché un squelette customisé.

Nous avons considéré les distances euclidiennes des points d'intérêts à l'origine et les angles dessinées par les membres de part et d'autre d'une articulation (épaule-coude-poignet par exemple) de ces articulations pour évaluer les erreurs de l'utilisateur par rapport à une référence donnée Or, par une méthode de Dynamic Time Warping , on peut évaluer à partir de tableaux de valeurs des positions des articulations prises à intervalle de temps régulier pour une dizaine de fois l'enchaînement sportif Les erreurs marginales de l'utilisateur pour une articulation en terme de distance et d'angle sont les suivantes :

$$E_{ponctuelle}^p(t) = |DTW_{distance}(seq_p(t), seq_{p, ref}(t))|$$

avec DTW_distance l'algorithme DTW modifié qui compare 2 séquences comportant les distances de chaque point d'intérêt p à l'origine à chaque instant(en terme de frame).

$$E_{jonction}^a(t) = |DTW(seq_a(t), seq_{a, ref}(t))|$$

avec DTW_angle l'algorithme DTW(voir annexe) modifié qui compare 2 séquences comportant les valeurs absolues des angles dessinés par 3 points d'intérêts contigus donnés par la formule suivante :

$$\theta_{jonction} = \text{Arcos}\left(\frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \cdot \|\vec{v}\|}\right)$$

où \vec{u} et \vec{v} sont les vecteurs représentant les membres contigus (le bras et l'avant bras relié par le coude par exemple)

seq_p et $seq_{p, ref}$ sont respectivement la séquence de l'utilisateur(en entrée du module classification) et une séquence dans la classe de mouvements(idéalement correspondant à l'exercice réalisé). On peut avoir 2 ou 3 références qui représenteraient les erreurs classiques. On considère un DTW normalisé par la somme des longueurs des 2 séquences par convention [1][2] avec une implémentation classique.

Et, on aurait à intervalle régulier en terme de frames une erreur globale mise à jour :

$$E_{globale}(t) = \sum_{p \in P} \left(\frac{E_{ponctuelle}^p(t)}{h} \right)^{\alpha_p} + \sum_{a \in A} (E_{jonction}^a(t))^{\alpha_a}$$

h est la taille de la personne. P est l'ensemble des points d'intérêts repérés par la Kinect, plus précisément des points d'intérêts dont dépendent le mouvement a priori, mais en première approximation dans nos programmes, on considéra tous les joints.

A est l'ensemble des jonctions suivantes :

- SkeletonPoints.SHOULDER_LEFT,SkeletonPoints.NECK,SkeletonPoints.SHOULDER_RIGHT;
- SkeletonPoints.SHOULDER_LEFT,SkeletonPoints.ELBOW_LEFT, SkeletonPoints.WRIST_LEFT;
- SkeletonPoints.SHOULDER_RIGHT,SkeletonPoints.ELBOW_RIGHT, SkeletonPoints.WRIST_RIGHT;
- SkeletonPoints.HIP_LEFT,SkeletonPoints.SPINE_BASE,SkeletonPoints.SPINE_MID;
- SkeletonPoints.HIP_RIGHT,SkeletonPoints.SPINE_BASE,SkeletonPoints.SPINE_MID;
- SkeletonPoints.KNEE_LEFT,SkeletonPoints.SPINE_BASE,SkeletonPoints.KNEE_RIGHT;
- SkeletonPoints.HIP_LEFT,SkeletonPoints.KNEE_LEFT,SkeletonPoints.ANKLE_LEFT;

- SkeletonPoints.HIP_RIGHT,SkeletonPoints.KNEE_RIGHT,SkeletonPoints.ANKLE_RIGHT.

et α_p et α_a des paramètres à fixer en fonction du point d'intérêt et de la jonction considérée. En effet, on pourrait atténuer l'erreur de certaines articulations avec ce paramètre.

On remarquera que, pour assurer l'homogénéité dans la formule précédente, on a normalisé par la taille , en supposant un modèle proportionnel entre l'erreur brute et la taille.

Ainsi on se retrouve avec des erreurs globales(normalisées puisqu'obtenu par DTW) par rapport à chaque référence, et on cherche le minimum de celles-ci pour déterminer la référence la plus proche, une référence étant une séquence de mouvement dans la base de données. Cette erreur globale nous permet aussi de savoir à quel point l' utilisateur se trompe dans son mouvement, et on devra fixer un seuil d'acceptabilité s_α du mouvement quand on compare à des références qui serait l'erreur minimale tolérable pour qu'une séquence de mouvements soit suffisamment stable pour correspondre à un élément de la classe de mouvement.

5.5.3. Motivation de la démarche

Cette démarche nous paraît être la bonne dans le sens où elle prend en compte le fait que l'enregistrement du squelette peut être en retard par rapport à celui de la référence, et que l'utilisateur n'aille pas au même rythme que l'utilisateur. Et on somme tout cela, sur un nombre de frames donné pour avoir une erreur globale pour un intervalle de temps donné. On prend en compte 2 façons de mesurer l'erreur avec chacune des avantages certains.

Celle par les distances semble assez naturelle et donne des erreurs finales du même ordre de grandeur que celle avec l' angle en supposant que l'on ait centré par rapport au centre du bassin le squelette rendu par la Kinect. Elle rend compte de plus de degrés de liberté que celle uniquement avec la détermination d'un angle. Elle rend mieux compte de toutes les erreurs, mais la méthode de l'angle semble rendre compte d'une erreur plus visible aux yeux d'un coach sportif, particulièrement pour des exercices sollicitant beaucoup les articulations. La méthode par les angles permet par ailleurs de ne plus avoir le désavantage de la taille et d'avoir a priori une indétermination sur les angles plus petite que sur les distances après normalisation. En effet, pour les erreurs ponctuelles ou les erreurs sur les jonctions, on a les indéterminations suivantes :

$$|dD| = \left| \frac{xdx + ydy + zdz}{\sqrt{D}} \right|$$

En supposant que l'on ait une longueur constante des membres en question(ce qui est à peu près vrai pour le bras/avant-bras et la cuisse/jambe)

$$|d\theta| = \left| \frac{dx' dy'}{r} \right|$$

avec : θ un angle décrit dans le plan(Ox'y') et r la longueur du membre. D'ailleurs en pratique, l'angle formé par 2 articulations contigues se trouvent souvent à peu près dans un des plans composés de deux axes du repère cartésien de la Kinect.

L'incertitude angulaire en ordre de grandeur serait plus petite a priori comme produit de deux écarts aux valeurs. Il faut néanmoins prendre en compte les limitations de la Kinect[3]. Or, on voit que les indéterminations de la profondeur sont dans certains cas plus grandes que la valeur même de la profondeur, donc notre hypothèse n'est pas tout le temps juste. Ainsi, il serait dans ces cas

préférable de déterminer l'erreur par la méthode des distances.

De plus, on a supposé que l'on avait que des points visibles alors qu'en pratique si on ne se place pas à la bonne distance, on peut avoir des points calculés et dans ce cas, l'indétermination est complètement aléatoire et peut-être grossière.

Remarques : Pour optimiser la classification, il faudrait se placer le plus près possible afin de minimiser l'incertitude sur la profondeur, mais on ne peut se placer à moins de 2 mètres, sinon la Kinect ne capte plus le squelette.

Par conséquent, pour rendre notre algorithme plus robuste, on a fait le choix de combiner les 2 méthodes pour calculer les erreurs.

5.5.4. Avancement

Démarche de tests

Nous avons réalisé une interface graphique qui affiche le squelette et nous permet de l'étudier sous différents points de vue, et de corroborer les résultats des tests. Nous avons pu stocker les données du flux Kinect sous la forme d'un fichier texte exploitable sans la caméra, et qui peut être relu. On a donc pu travailler à plusieurs dessus.

Nous avons d'abord testé nos programmes en appliquant DTW à deux séquences de coordonnées de chaque articulation durant toute la session de l'exercice. On ne fera pas, dans un premier temps, la mise à jour temporelle des erreurs, car d'emblée on réalise des enregistrements assez courts, et de plus, on veut simplement tester qualitativement si l'erreur obtenue est cohérente avec les données. En effet, l'idée est d'obtenir un seuil d'acceptabilité s_a d'un mouvement par rapport à une référence. A priori, on peut passer aisément de ce plan de tests à la mise en place de notre démarche. Il faudrait gérer les appels à la méthode DTW durant la session d'exercice, et mettre à jour l'erreur.

Voici comment on procède pour les tests : A partir de la Kinect et des méthodes héritées de la bibliothèque J4K, on récupère deux tableaux contenant pour chaque articulation une séquence temporelle de coordonnées 3D de chaque point d'intérêt. On compare les séquences temporelles associées des 2 tableaux à l'aide d'une méthode DTW qui prend en entrée 2 tableaux de flottants représentant les 2 séquences temporelles à comparer. Puis on applique la démarche globale, en prenant $\alpha_p = 1$ et $\alpha_a = 1$.

Pour les 2 premiers tests, avant le PAN2, on ne fait intervenir que le terme dans l'erreur liée à la distance. Et on obtient donc une erreur dite finale(normalisée), puisqu'on a dû étudier les impacts de normalisations variées en faisant les 2 premières batteries de tests.

Finalement, on reste sur une normalisation usuelle.

Le code Java principalement utilisé est sur la branche Christophe_Classification_PAN3 de gitlab pour la version la plus récente.

Plan de tests

On teste pour la flexion-extension des jambes dans différentes configurations bras droit levé, 2 bras

levés, main droite sur la tête et le modèle de base, c'est-à-dire que l'on effectue les exercices en maintenant à un certain moment ces configurations au lieu de ce qui est établi comme référence. Une seule personne effectue tous les mouvements pour ne pas rencontrer le problème de la différence de taille. On s'est assuré d'avoir un environnement avec la luminosité adéquate, et de bien se placer en face de la Kinect, ce qui permet d'avoir le squelette le plus pertinent possible, et de pouvoir avoir à la relecture un squelette qui nous donne suffisamment d'information. Ce test doit nous permettre de vérifier autrement que par la visualisation sur l'interface graphique les différences entre les séquences. La session a une durée en nombre de trames. On peut par la suite diminuer le temps de session.

Voici les résultats :

1ère batterie de tests (avant PAN2) :

Pour une normalisation de DTW_distance par rapport à la longueur de trames la plus grande, on obtient à titre qualitatif :

Flexion des jambes : comparaison entre la référence et différentes configurations :

Configuration du test	Erreur finale	Erreur totale	Nombre de trames	Commentaire
main droite sur la tête	0.34	151.04	440	Différence grossière en théorie, ordre de grandeur par rapport aux autres erreurs vérifié en pratique.
bras droit levé	0.29	213.63	728	Une erreur pas très importante, ordre de grandeur par rapport aux autres erreurs vérifié en pratique.
2 bras levés	0.60	518.72	863	Différence non négligeable, ordre de grandeur par rapport aux autres erreurs vérifié en pratique.

A première vue, la cohérence des erreurs est respectée de façon assez générale, , mais on n'obtient certaines incohérences, et pour chaque comparaison des valeurs anormalement élevées comme le montrent les rapports d'erreurs fichier txt pour les articulations suivantes le cou(2), le coude gauche(5),l'épaule droite(8), la cheville gauche(14) et le genou droit(17) qui semblent être des membres instables lors de la session visibles à l'interface graphique. Nous avons ainsi par rapport à notre démarche pris $\alpha_p = 1$, pour tout point d'intérêt autre que ces articulations pour faire notre tableau.

On obtient une erreur finale de 0.33 entre la flexion bras droit levé et la flexion deux bras levés.

En effet, on obtient bien en normalisant l'erreur, que le mouvement bras droit levé est le plus proche de la référence, et le mouvement main sur la tête est le plus éloigné.

Les valeurs obtenues semblent suivre une certaine tendance, mais étrangement l'enregistrement qui est censé correspondre à la référence donne une erreur qui s'élève à 0.50, ce qui est anormalement élevé. Après coup, on a remarqué à l'interface graphique, des différences de placement assez grandes qui pouvaient être visible à l'oeil nu.

L'erreur semble liée au fait que lors de la comparaison, du fait des différentes conditions d'enregistrement, ou encore du fait que les axes pour les coordonnées ne soient pas les mêmes, ce qui est un peu visible lors de la relecture à l'inteface graphique. Il faut donc a priori recentrer. Mais comme on peut le voir sur les images de l'interface graphique, le recentrage nécessaire est assez minime pour nos tests suivants.

2ième batterie de tests(avant PAN2)

On considère une normalisation de DTW_distance par rapport à la longueur de trames la plus grande

Cette fois, on n'a pas eu besoin de changer la valeur du paramètre α_p car on obtient des erreurs d'articulation relativement stables (cf les rapports d'erreur sur gitlab branche Christophe_Classification_PAN3), même si les points d'intérêts douteux cités précédemment ont des valeurs DTW un peu plus élevées(ce qui peut être justifié par le fait que ces articulations sont assez sollicitées dans le mouvement). D'ailleurs, de façon générale, on obtient des résultats de valeurs finales de l'erreur plus basses que précédemment.

Flexion des jambes complète : comparaison entre la référence et différentes configurations pour un peu plus d'une dizaine de fois cet exercice :

Configuration de test	Erreur finale	Erreur totale	Nombre de trames	Commentaire
mouvement bien exécuté	0.14	109.06	794	Pas de différence en théorie, erreur finale minimale en pratique.
main droite levée	0.13	116.13	856	Une erreur anecdotique, erreur finale minimale en pratique.

Configuration de test	Erreur finale	Erreur totale	Nombre de trames	Commentaire
sans aller complètement de haut en bas	0.26	246.18	856	Une erreur a priori grossière, établit une estimation du seuil d'acceptabilité au delà duquel on a des taux d'erreurs majeures , car valeur qui double par rapport au taux d'erreur précédent.

Levée des jambes debout : comparaison entre la référence et différentes configurations pour pas loin d'une quinzaine de fois cet exercice:

Configuration de test	Erreur finale	Erreur totale	Nombre de trames	Commentaire
jambe droite levée à mi-hauteur	0.17	144.29	835	Erreur non négligeable mais pas répréhensible, on obtient bien une erreur bien supérieure aux erreurs pour la flexion complète bien faite en pratique
buste baissé	0.36	319.04	806	Une erreur assez grave sportivement, bien retranscrit par le taux d'erreur, mais on peut corriger.

Les résultats sont assez probants, d'autant plus que les taux d'erreur entre une flexion et une levée de jambe s'élève à 0.62 et 0.66.

Bilan(avant PAN2)

Donc, d'après nos tests, notre système devrait pouvoir reconnaître quel est le type d'exercice pratiqué par l'utilisateur, et il devrait pouvoir déterminer 4 types d'erreurs :

- les erreurs anecdotiques avec un plafond aux alentours de 0.14 ou 0.15
- les erreurs légères avec un seuil de taux d'erreur aux alentours de 0.20

- les erreurs graves qui dépassent 0.20 et varient de l'ordre de 0.40 ou plus (on y compte aussi l'erreur sur le type d'exercice).

On pourra d'ici-là peut-être prendre en compte que les positions calculées pour les calculs d'erreur, car elles sont plus fiables que les autres. On pourrait donc soit enlever certaines articulations dans la détermination de l'erreur, soit minimiser l'erreur ponctuelle (plafond). L'idée générale est de, si possible, ne pas faire varier l'erreur globale trop rapidement si on a peu de grosses variations ponctuelles.

Test final(avant PAN3)

On a testé divers algorithmes utilisant la méthode uniquement par calcul d'angles avec des résultats relativement bons par rapport à notre objectif, mais qui ne sont peut-être pas satisfaisants d'un point de vue physique. En effet, le maximum de variations d'angle ne correspond pas à l'endroit où l'erreur de mouvement a été commise dans la plupart de nos tests. On s'est donc rendu compte que les erreurs se répercutaient sur tout le corps, en particulier avec la méthode des distances.

On est finalement arrivé à la démarche globale qui prend en compte aussi la distance avec laquelle on avait une bonne cohérence par rapport aux erreurs ponctuelles.

Batterie de tests n°3 :

Ce test reproduit assez bien la situation que l'on va présenter au PAN3. On reprend ce qu'on a proposé dans la démarche globale et la démarche de tests

Ici, on a fait réaliser des exercices et ses erreurs associées par 2 personnes de taille différente, ce qui fait que l'on peut avoir des mouvements à comparer. En effet, on prend la séquence de la première personne et on réalise la classification avec la classe qui contient les séquences de la deuxième personne.

On s'est assuré de faire le même nombre de mouvements d'une personne à l'autre. Pour la plupart des mouvements, on a fait 7 enchaînements, et 10 lorsqu'on réalise des exercices qui nécessite de alterner droite gauche.

Lorsqu'on entre un mouvement d'une personne dans la classification, on compare avec la classe de l'autre personne. On s'assure ainsi de ne pas avoir une erreur nulle, et on étudie aussi la validité de notre algorithme quel ce soit la taille.

Le détail des tests est dans la branche Christophe_Classification_PAN3, dossier Classification workspace, dossier tests et rapports de tests. Il y a aussi les classes de mouvements considérées :

Différentes configurations(Id) pour les 4 exercices(N°classe) au PAN3:

Id\N°classe	1 : Flexion des jambes	2: Fentes	3 : Montée des 2 genoux	4 : élongation bras	5 : Travail sur les genoux
0	référence(7)	référence	référence rythme lent(10)	référence	référence(10)

Id\N°classe	1 : Flexion des jambes	2: Fentes	3 : Montée des 2 genoux	4 : élongation bras	5 : Travail sur les genoux
1	flexion d'une jambe(7)	mouvement de haut en bas, pas de l'arrière vers l'avant	rythme rapide(10)	//	2 genoux pliés(14-16)
2	avec buste pas droit(7)	dos non courbé, regard loin devant	un genou monté(10)	//	que d'un côté(14)
3	pas jusqu'au bout(7)	talon de la jambe arrière décollé et part légèrement sur l'extérieur	pieds levés à la place des genoux(14)	//	//
4	avec les jambes trop serrées(10)	pointe de la jambe avant légèrement vers l'intérieur	dos penché(10)	//	//
5	avec les jambes trop écartées(10)	//	genoux montés pas assez haut	//	//

Le système que l'on a prévu marche à 100% pour ces séquences. Nous avons fait les calculs des erreurs finales et on les a comparées entre elles pour déterminer le minimum, et donc le plus proche voisin dans la classe de mouvements de l'exercice réalisé.

Après quelques tests sur tous nos mouvements, le seuil s_α a été fixée à 3, ce qui paraît assez important, mais il est assez raisonnable de le prendre relativement élevé, car notre coach sportif est déjà suffisamment exigeant pour que la stabilité pénalise encore plus l'utilisateur.

5.5.5. Eléments d'intégration

La classification a été intégrée au niveau de la méthode `onSkelonFrameEvent` issue de la bibliothèque J4K qui récupère les coordonnées 3D de la personne face à la Kinect tout au long de l'utilisation. Le traitement de classification compare alors à la classe de mouvements transmis par le serveur, ce qui va se traduire par un message qui va être reçu par le client côté Android.

La liste de tous les mouvements finalement retenus se trouve dans le rapport Android.

5.5.6. Annexe

Un pseudo-code du DTW utilisé :

```

float DTW(array [1..n], array [1..m]) {
    DTW := array [0..n, 0..m] //2 séquences de longueur n et m qui selon les cas représentent des coordonnées ou des angles
  
```

```

for i := 1 to n
    DTW[i, 0] := infinity
for i := 1 to m
    DTW[0, i] := infinity
DTW[0, 0] := 0

```

```

for i := 1 to n
    for j := 1 to m
        cost := d(s[i], t[j])
        DTW[i, j] := cost + minimum(DTW[i-1, j ],      // insertion
                                      DTW[i , j-1],      // deletion
                                      DTW[i-1, j-1])     // match

```

```
return DTW[n, m]/(n + m)
```

5.5.7. Bibliographie spécifique

- [1] Dynamic Type Warping for Gene Expression Time Series, Elena Tsiporkova, slide 11
- [2] <http://luscinia.sourceforge.net/page26/page14/page14.html> (à propos de la normalisation)
- [3] <http://iut-gmp.univ-lille1.fr/fichiers/LPVI/RapportFinalkinect.pdf>, Bertrand PECUCHET (page 13)

5.6. Kinect

5.6.1. Description des objectifs

Expert du module : Jean Le Feuvre (jean.lefeuvre@telecom-paristech.fr)

- Comprendre le principe de base du fonctionnement de la Kinect
- Mettre en place la communication entre une Kinect v1 et un PC
- Capturer des mouvements et les enregistrer
- Formater les données
- Étudier les limites de la Kinect et en rendre compte dans le prototype final

5.6.2. Enregistrement des mouvements

Nous avons utilisé la librairie J4K (Java For Kinect) pour communiquer avec la Kinect branchée sur le port USB 2.0 d'un PC tournant sous Windows après avoir installé la SDK 1.8 de Kinect for Windows.

À chaque frame de mouvement détectée par la Kinect, l'ordinateur la traite pour finalement, à la fin de l'enregistrement, sauvegarder la séquence de frames (ce que nous appellerons le *mouvement*) dans un fichier texte. Pour chaque frame, nous nous intéressons à 20 point d'intérêts (sur les 25 que

propose la Kinect). Pour chaque point d'intérêt, la Kinect fournit trois coordonnées correspondant à la position dans l'espace de celui-ci, ainsi qu'un booléen indiquant si le point a été effectivement détecté par la Kinect, ou si sa position a été calculée (algorithme interne au dispositif). Ce dernier point est important à noter car nous nous y sommes intéressés pour apprécier la qualité des enregistrements (se référer au rapport de tests Kinect ci-après).

Le fichier texte d'un enregistrement de *mouvement* est formaté de la façon suivante :

mouvement.txt

Enregistrement d'un squelette Kinect

[x0,y0,z0,x1,y1,z1,...,x19,y19,z19]

[b0,b1,...,b19]

...

...

[x0,y0,z0,x1,y1,z1,...,x19,y19,z19]

[b0,b1,...,b19]

Ainsi, chaque frame donne lieu à deux lignes dans le fichier texte de l'enregistrement. La première contient les positions dans l'espace des 20 points d'intérêts auxquels nous nous intéressons. La deuxième contient les booléens mentionnée ci-dessus : il vaut 2 si le point était visible, 1 si la position du point a été calculée.

Des exemples de fichiers d'enregistrements sont disponibles dans le dépôt.

Ces fichiers sont ensuite exploités par le module classification, de manière à comparer le mouvement enregistré en temps réel par la Kinect lorsque l'utilisateur de SportBot réalise un exercice avec les mouvements correct et incorrects préalablement enregistrés par l'équipe de développement. Ils sont également exploitables par un soft (cf. module *Synthèse 3D*) affichant un avatar virtuel permettant de relire la séquence de frames.

5.6.3. Rapport de tests Kinect

Généralités

Lors des premiers tests d'enregistrement de séquences Kinect, nous avons fait quelques observations sur les résultats en relisant les séquences (grâce au code fourni permettant d'afficher le squelette à partir des fichiers contenant les séquences de positions).

Nous avons vite remarqué que des conditions devaient être respectées pour que les positions enregistrées par la Kinect soient fidèles à la réalité, mais de manière plus rudimentaire pour que la Kinect détecte un squelette. Par exemple, une distance trop grande ou au contraire trop courte au capteur semblait compromettre ce dernier point.

Nous avons également observé que, à une distance donnée, des mouvements trop amples (par exemple avec les bras levés) avaient pour conséquence des positions erronées pour les points "extrêmes". Cela traduit une certaine "tolérance" en ouverture angulaire de la Kinect.

L'objectif des tests que nous avons mené est donc d'essayer de quantifier ces zones de tolérance et leur importance est capitale, puisque cela aura des conséquences immédiates sur la suite du projet :

certains mouvement que nous avions prévu d'intégrer au programme d'entraînement de SportBot pourraient se retrouver disqualifiés s'ils s'avèrent être incompatibles avec les conditions d'optimalité mises en avant par ces tests.

WARNING

Bien que nous ayons essayé d'obtenir des résultats quantitatifs, nous n'avons pas eu d'autre choix que de définir certains seuils *arbitrairement*, notamment en ce qui concerne la stabilité des points. Les mesures (distance à la Kinect, hauteur) ont été effectuées à l'aide d'un mètre.

Remarque : Nous avons remarqué que la Kinect a du mal à repérer le squelette lorsque l'enregistrement débute trop proche. Par exemple, en commençant l'enregistrement à 1,50m de la Kinect, aucun squelette n'est détecté. La Kinect le détecte pourtant si on effectue par exemple un saut (sûrement est-ce dû au fait que les pieds, qui étaient "hors champ" sont alors détectés par la Kinect qui peut ensuite deviner leur position dans la suite de la séquence). En revanche, si on commence l'enregistrement dans la "zone de confort" (par exemple à 2,50m de la Kinect) puis que l'on se rapproche, le squelette est toujours détecté par la Kinect (même si les positions de certains points d'intérêt, notamment la tête et les pieds, peuvent devenir hasardeuses).

Prémière série de tests : détermination d'une plage de distance

Présentation des tests

L'objectif est de déterminer une plage de distances à la Kinect $[D_{min}; D_{max}]$ pour lesquelles le squelette enregistré est stable. Pour ce faire, on se place dans les conditions suivantes : - ni le modèle humain ni les habits qu'il porte ne changent pendant le test ; - l'éclairage est fixé : lumière naturelle (temps dégagé en milieu d'après midi) ; - la hauteur de la Kinect est fixée : $H_{Kinect} = 81cm$

On effectue ensuite un enregistrement de 10 secondes pour différentes positions, en partant de 3,50 m. Pour pouvoir comparer objectivement les résultats, le même mouvement a été effectué pour chaque enregistrement (un balancement sur les jambes, ainsi qu'avec les bras, face à la Kinect).

Pour chaque enregistrement, il y a bien sur une observation qualitative permettant de juger de la qualité apparente de l'enregistrement. Cependant, pour essayer de tirer des informations plus quantitatives des enregistrements, nous avons comparé le nombre de points (articulations) dont la position a été calculée par la Kinect au nombre de points ayant été catégorisé comme "visible" par la Kinect. Le calcul de la proportion de points visibles sur le total de points est effectué automatiquement sur un fichier enregistré grâce à la méthode **showStats(String filename)** contenue dans le fichier **StatsPoints.java**.

NOTE

Les enregistrements de ces tests sont disponibles dans le dossier `./tests/distance/`

Résultats

Premier modèle : Axel (taille : 183 cm)

Les enregistrements de ce test sont disponibles dans `./tests/distance/axel/`

Table 1. Satisfaction de l'enregistrement en fonction de la distance à la Kinect

Distance (cm)	Proportion de points visibles	Commentaires
350	98,4%	Bon enregistrement, quelques "sauts" dans la position de certains points mais cela reste exceptionnel.
275	99,8%	Bon enregistrement, fidèle à la réalité. Quelquefois, la position des mains et des pieds paraît un peu hasardeuse.
230	99,6%	Bon enregistrement, fidèle. La position des mains et des pieds semble plus fidèle à la réalité.
185	89,5%	La position de certaines articulations n'est plus fidèle à la réalité. La tête grossit de temps en temps (la coordonnée de profondeur n'est pas correcte).
175	87,0%	On devine encore le mouvement, mais seulement pour les bras. Certains enchaînements sont incohérents. L'enregistrement n'est clairement pas satisfaisant.
150	47,3%	L'enregistrement n'est pas du tout exploitable. De plus, la Kinect a enregistré très peu de points (environ trois fois moins que pour les enregistrements précédents).

Second modèle : Christophe (taille : 170 cm)

Les enregistrements de ce test sont disponibles dans ./tests/distance/christophe/

Table 2. Satisfaction de l'enregistrement en fonction de la distance à la Kinect

Distance (cm)	Proportion de points visibles	Commentaires
350	96,2%	Bon enregistrement, bien que les pieds et les mains soient un peu "sécoués".

Distance (cm)	Proportion de points visibles	Commentaires
275	99,4%	Bon enregistrement, le problème sur les pieds et les mains est encore présent mais s'est arrangé par rapport à 350 cm.
175	99,5%	Enregistrement satisfaisant dans l'ensemble, mais la tête n'est pas très stable au début et la taille des jambes ne paraît pas réaliste (problème de détection des pieds ?)
165	70.6%	Il y a clairement un problème au niveau de la partie basse du corps (pieds et bas des jambes). Le reste du corps est fidèle.
160	58.2%	Quelques problèmes au niveau du bas du corps, mais mieux que pour 165 cm.
150	45.8%	Déformation du corps, points mal placés. Enregistrement clairement pas satisfaisant.

5.6.4. Analyse des résultats, conclusion

Bien que les résultats varient, on s'aperçoit que les résultats sont satisfaisants (tant au niveau de la proportion de points visibles par la Kinect qu'au niveau des observations qualitatives sur la qualité de l'enregistrement) pour une distance à la Kinect variant entre 1m75 et 3m50, en étant indulgent.

La plage de distance recherchée est donc environ l'intervalle [$D_{min} = 1,80m$; $D_{max} = 3,30m$].

Deuxième série de tests : détermination de l'ouverture angulaire

Présentation des tests

Dans la première série de tests, nous avons remarqué que c'était souvent les points extrêmes, en haut et en bas, qui posaient problème dans l'enregistrement. L'objectif de cette deuxième série de tests est donc de déterminer une ouverture angulaire pour le capteur Kinect, dans laquelle les enregistrements sont satisfaisants. Nous choisissons six distances à la Kinect différentes dans la plage de distances que nous venons de déterminer (et un peu autour), puis pour chaque distance D , nous cherchons à mesurer la hauteur H_{max} à partir de laquelle la Kinect ne détecte plus correctement la position de la partie la plus haute du corps (souvent un bras ou une main).

On remonte ensuite à l'angle (demi-ouverture angulaire) grâce à la formule suivante :

$$\tan(\theta) = \frac{H_{max} - H_{Kinect}}{D}$$

La hauteur de la Kinect est toujours fixée à $H_{Kinect} = 81cm$.

NOTE Les enregistrements de ces tests sont disponibles dans le dossier ./tests/angle_lim/

Résultats

Distance D (cm)	H_{max}	\tan(theta)	\theta (°)
350	260	0,511	27,1
300	230	0,497	26,4
250	205	0,460	26,4
200	190	0,545	28,6
175	170	0,509	27,0
150	142	0,401	22,1

5.6.5. Analyse des résultats, conclusion

La demi-ouverture angulaire moyenne mesurée est :

$$\theta_{moy} = 26, 3^\circ$$

avec un écart-type $\sigma = 2, 2^\circ$.

L'écart-type étant relativement peu élevé, les mesures nous donnent donc une bonne idée de la valeur de l'ouverture angulaire du capteur qui est d'environ 2θ .

Si l'on souhaite que la Kinect enregistre correctement les mouvements, il faudra donc trouver un **compromis** entre la hauteur de cette dernière et la distance à laquelle on effectue les mouvements, car une relation relie désormais les deux :

$$D_{min} = \frac{H_{Kinect}}{\tan(\theta)}$$

5.6.6. Annexe : recherche bibliographique

Dans le cadre de notre projet SportBot, nous allons enregistrer les mouvements effectués par l'utilisateur pendant une séance d'entraînement sportif grâce à la technologie Kinect qui produit un squelette. Ce dernier sera ensuite enregistré et analysé (cf. *module classification*) pour en déduire le mouvement le plus proche correspondant parmi une base de données de mouvements pré-enregistrés contenant à la fois les *bons* mouvements et les *mauvais* mouvements.

Cette approche paraît simple mais occulte une difficulté assez majeure : la base de mouvements pré-enregistrés n'aura pas été effectuée à partir d'un enregistrement du squelette de l'utilisateur, mais au moment de la conception du logiciel. Or, les différences sont innombrables au sein d'une même espèce, et l'être humain est bien sûr loin d'échapper à la règle. Au-delà des différences de taille, les morphologies sont très différentes et il en est de même pour le squelette et la taille des os. Les squelettes de deux personnes faisant la même taille présentent parfois de fortes disparités. De

même, la taille des os n'est pas proportionnelle à la taille "globale". Nous citerons par exemple le nageur Michael Phelps et Hicham El Guerrouj, le record du monde du 1500 m en plein air et en salle, dont les jambes font la même taille alors que 18 cm les séparent lorsqu'ils se mettent dos à dos.

Dès lors, nous pouvons nous interroger quant à la généralisabilité de notre modèle. Quelle taille, quel type de morphologie devrions-nous choisir pour enregistrer nos mouvements dans la base de données afin d'être au plus proche de la morphologie d'un maximum d'utilisateurs ?

Comme nous ciblons dans le cadre du PACT une population plutôt locale, nous souhaiterions définir ce qu'est la morphologie standard d'un Français. Il est inutile de souhaiter réduire davantage l'espace géographique compte-tenu des fortes migrations internes au pays qui rendrait par exemple peu pertinente la recherche d'une définition d'un Francilien standard.

L'Institut Français du Textile et de l'Habillement (IFTH) fournit des données morphologiques [1] grâce à des mesures effectuées sur un échantillon de plusieurs milliers d'hommes français en 2003. Bien entendu, les différences sont conséquentes : 19,5% d'entre eux sont classés comme « petits » et mesurent 1,65m en moyenne, alors que les « très grands » représentant 8,5% de l'échantillon mesurent en moyenne 1,91m. La taille moyenne mesurée sur l'échantillon était de 175,6m, pour un poids moyen de 77 kg environ. Ces données sont confirmées par un article de *Science et Vie* [2].

La suite des recherches bibliographiques sur le sujet s'est avérée délicate : les sources des données sont parfois vaguement citées et un bon nombre d'études porte sur d'autres populations que celle ciblée initialement. Notamment, la NASA a effectué un grand travail de recherche sur le corps humain notamment en ce qui concerne sa taille, sa posture, sa surface et sa masse, par exemple dans un souci d'intégrabilité dans les espaces destinés à accueillir un équipage. [3]

Les tables anthropométriques comme celle qui suit permettent d'avoir une idée des tailles moyennes (ou plutôt des proportions moyennes, relativement à la taille totale). Elles sont souvent accompagnées de schémas permettant de visualiser les proportions indiquées. Cependant, une fois encore les variations sont bien trop importantes d'un individu à l'autre comme nous l'avons illustré précédemment avec l'exemple des sportifs Michael Phelps et Hicham El Guerrouj. De plus, les études sont trop peu nombreuses et réalisées sur des échantillons insuffisants pour espérer en tirer des conclusions satisfaisantes.

Longueur (%)	Typique	Exemples
Partie du corps	Homme	Homme
Tronc	30	54,0
Tête et cou	13,8	24,8
Cuisse	23,2	41,8
Jambe	24,7	44,5
Pied	4,2	7,6
Bras supérieur	17,2	31,0
Avant-bras	15,7	28,3
Main	10,4	18,7
Total	100,0 %	180 cm

Figure 1. Table anthropométrique établie par Paolo de Leva

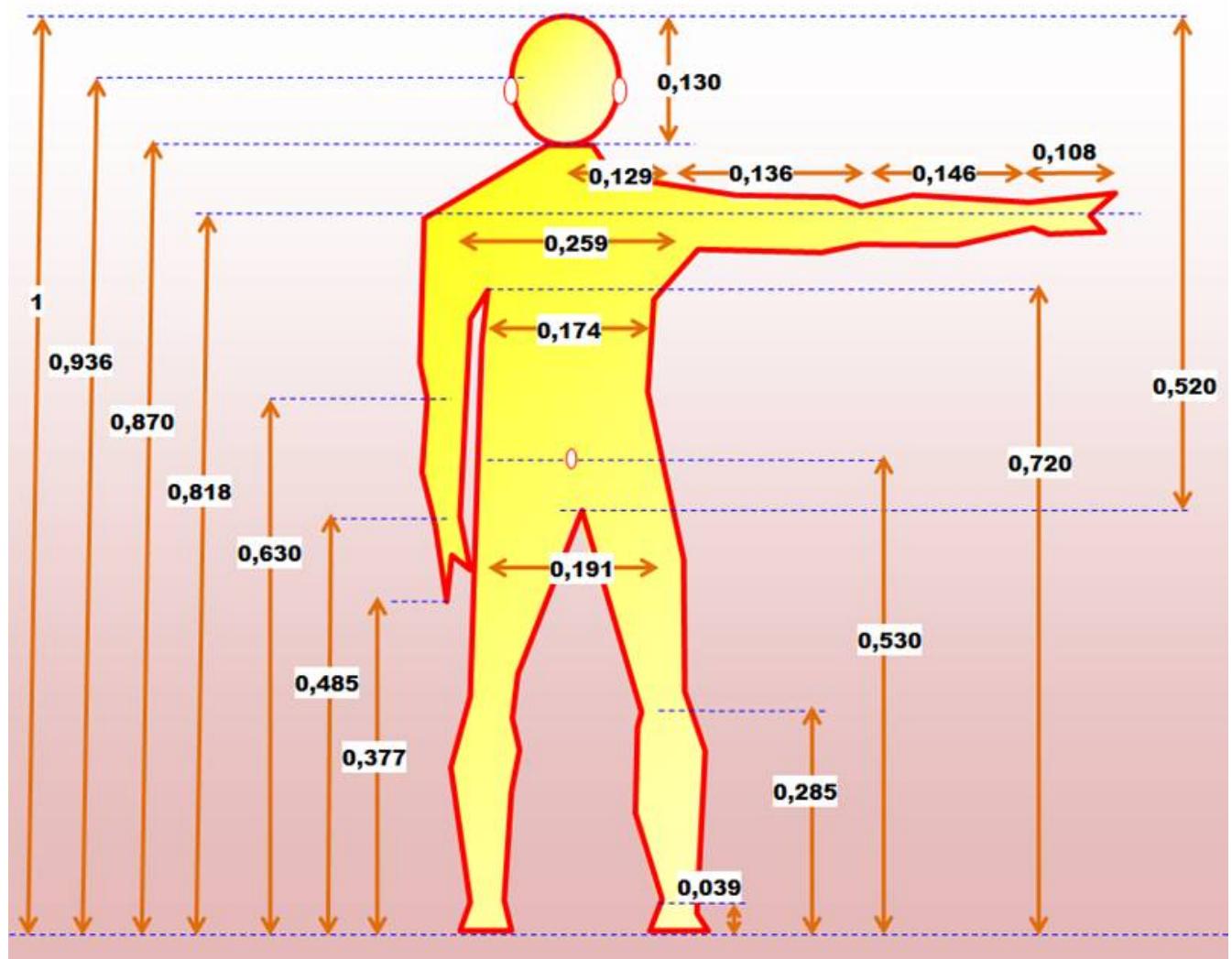


Figure 2. Schéma d'anthropométrie [5]

Pour finir, en ce qui concerne la diversité des morphologies au sein de la population et notamment des éventuels utilisateurs de SportBot qui ne seront évidemment pas tous un "Français standard", il serait toujours possible de pallier (du moins partiellement) ce problème en faisant enregistrer chaque mouvement par une grande diversité de performeurs permettant de couvrir la plus grande proportion possible de la population. La difficulté sera alors d'élaborer un processus permettant d'identifier facilement la morphologie la plus proche de celle de l'utilisateur.

- Références :
- [1] <http://www.doctissimo.fr/html/sante/mag/2006/9331-mensurations-francais.htm>
 - [2] *Pourquoi les femmes sont-elles plus petites ?*, février 2017 <https://www.science-et-vie.com/corps-et-sante/pourquoi-les-femmes-sont-elles-plus-petites-7885>
 - [3] NASA, *Man-system integration standards, Volume 1, Section 3.* pp. 32-79; NASA-STD-3000 275e <https://msis.jsc.nasa.gov/sections/section03.htm>
 - [4] *Proportion du corps humain*, Gérard Villemin <http://villemin.gerard.free.fr/Biologie/CorpsPro.htm>
 - [5] <https://www.houzz.fr/photo/1667791-anthropometry>

5.7. Android

5.7.1. Réalisations PAN 2

Pour le PAN 2, nous avons réalisé le squelette de l'application Android, et la gestion des informations de l'utilisateur (prénom, nom, âge, goûts musicaux).

L'application comporte 5 activités :

- Le menu d'accueil
- La page d'entrée des informations
- La page pour choisir l'utilisation de la Kinect (oui/non)
- Le menu principal avec la future 3D, et un bouton de réglage
- Une page de réglage pour modifier les informations utilisateur, activer ou non la Kinect, et utiliser la communication client/serveur (WIP)

Pour la partie informations utilisateurs, nous avons mis en place des Shared Preferences qui permettent de stocker des informations telles que String, booléen, entiers dans le téléphone. De cette façon nous pouvons récupérer ces données dans le menu des réglages si jamais l'utilisateur veut modifier ses informations.

Pour les goûts musicaux, nous avons créé un dialogue personnalisé afin de combiner correctement Shared Preferences et Checkbox.

5.7.2. Réalisations au PAN 3

Dans ce qui suit, on suppose que l'on a déjà une intégration d'une version fonctionnelle de la classification.

La première phase du développement est passé par une factorisation des préférences utilisateur. Nous avons factorisé les goûts musicaux afin de les stocker dans un array qui fait correspondre goût avec booléen.

Nous avons ensuite intégré la 3D dans l'activité principale, développée par le module synthèse 3D.

Nous avons créé une classe BubbleMessage qui permet d'afficher des bulles de messages sur l'écran. Ces bulles sont censées sortir de la tête du robot afin de le faire parler. Chaque bulle a des attributs de coordonnées. Une bulle a deux méthodes principales :

open qui démarre une animation ouverture de la bulle (un AnimationSet avec un fade et une scaleAnimation), et une animation de fermeture (l'inverse).

Une classe TrainingBuilder permet de récupérer des données sur les exercices à faire par l'utilisateur stockés dans des objets Exercice (avec des informations pour obtenir la classe de mouvement correspondante et la durée de l'exercice).

A partir de cette classe et d'une classe pour récupérer les informations d'une classe de mouvement, on montre à l'utilisateur les mouvements qu'il doit effectuer.

Les messages s'affichent lorsque l'on reçoit des informations de la classification, en utilisant la classe BubbleMessage et une classe BubbleMessageManager qui permet de gérer l'arrivée de nouveaux messages pour la classe BubbleMessage.

5.7.3. Réalisations au PAN 4

Implantation de la synthèse vocale

On a pris une synthèse vocale pré-implémentée que l'on lancera lors des transitions entre les exercices d'un programme. Il y a a priori plusieurs messages pour la synthèse vocale selon le mode de sévérité de SportBot. Ces modes ont été implémentés en tant que boolean isRude, et il apportera donc une différence qu'au niveau des directives prononcées par la synthèse vocale. Ces modes seront donc proposés en début de séance via un QCM configuré en début de séance.

Mise en place du QCM(intégré par le module Intégration&tests)

Nom, sexe (pas spécifiquement besoin), poids, taille,

Fais-tu souvent du sport ? Oui, Non

(optionnel) Si oui plutôt quel type de sport ? Sport d'équipe, individuel, à haute sensation

Qu'aimes-tu plutôt dans le sport ? La compétition, l'esprit d'équipe, le bien-être, la relaxation, une forme d'expression qui te permet de te libérer et de te montrer aux autres

Aimes-tu écouter de la musique pendant le sport ? Oui, non, zut

Quelle est ta motivation pour mener à bien ce programme sportif ?

Souhaites-tu dépasser tes limites ou préfères-tu te contenter de simplement remplir les objectifs ? influencera la “dureté” de SportBot en ce qui concerne l’intonation et l’intensité des exercices ou du nombre de répétitions

Aimerais-tu que je communique avec toi durant le parcours du combattant qui t’attend ? Oui, Non, je me débrouille

Ne t’inquiète pas les séances vont être très agréables.

De la musique pendant la séance ? Oui/non

Tu peux choisir un programme(mode pour commencer) ...

Au programme :

- flexion jambes
- fentes
- deltoïdes
- montées de genoux
- étirement latéral bras droit, bras gauche
- étirement croisé bras droit, bras gauche
- torsions

Un programme regroupera plusieurs de ces exercices, obtenu avec l’outil TrainingBuilder.

Intégration d’un décor pour le robot + détails 3D

Amélioration des bulles (couleurs, tailles en fonction du message)

Implémentation d’une musique qui se lance au lancement de l’application (pour créer l’ambiance)

Implémentation d’un lecteur de musique propre à l’application, qui lance des playlists en fonction des choix de goûts de l’utilisateur

- L’utilisateur pourra changer de playlist ou de goût musical durant l’entraînement.
- L’utilisateur peut aussi choisir de jouer sa propre musique (stockage du téléphone)

5.7.4. Ressources

- La synthèse vocale de Google TTS(talk to speech)

5.8. Intégration et tests

Dans ce qui suit, nous avons fait le choix de décrire de façon chronologique ce qui a été fait étape par étape.

5.8.1. Réalisations PAN 3

Voici le détails du module Intégration et tests.

Nous avions fini avant le PAN3 une implémentation fonctionnelle de la classification, du soft synthèse 3D. Voici les tests initiaux pour les intégrer au projet pour la partie création de base de moves et évaluation des mouvements d'un utilisateur par rapport à un mouvement référence (se référer aux modules respectifs pour plus de détails)

1.Kinect allégée + Classification Cad : kinect pour lire le squelette transmis par la Kinect et transmettre le stream de squelette sous la forme de 2 tableaux pour chaque frame : [positions[0,x],positions[1].y,...,positions[20].z] //tableau de 60 flottants, pour les 20 positions, 3 axes [visible[0],...,visible[20]] //visible[i] vaut 1 si la position est calculée par la Kinect, 2 si elle est visible par la Kinect (\Rightarrow plus fiable) Ces 2 tableaux sont envoyés au module classification qui les lira via une méthode static `readFromKinect(float[] positions, int[] visibles)`, vérifiera que l'input est correct (positions de longueur 60 et visibles de longueur 20 (chaque élément de valeur 1 ou 2)), affiche un message d'erreur sinon. Le module classification traite alors ses données en fonction du mouvement auxquelles elles doivent correspondre (via un id de mouvement transmis au module via une interface pour indiquer quel mouvement choisir dans la base de données, qui sera remplacé par un id envoyé par le smartphone lors de l'implémentation de la communication client-serveur). Input : mouvements de l'utilisateur en fonction d'un mouvement à exécuter, id du mouvement à exécuter à entrer pour le module classification. Output : identification de l'erreur commise par l'utilisateur qui lui est transmise, ou simplement lui transmettre que le mouvement est bien exécuté Le module de classification étant supposément déjà testé (cad que la classification fonctionne correctement, et suffisamment rapidement pour être effectuée en condition réelle), utiliser les mouvements utilisés pour les tests de la classification lors de cette phase de test

2.Kinect allégée + Synthèse 3D allégée Cad : Kinect pour lire (depuis un PC ou depuis la Kinect) ou sauvegarder un squelette, ajout d'une surcouche pour découper les squelettes enregistrés entre 2 moments pour enregistrer des squelettes qui sont constitués d'un même nombre de mouvements Module synthèse 3D qui permet de générer un mouvement moyen à partir de plusieurs mouvements fournis Ajout d'un petit utilitaire qui permettra de fusionner plusieurs mouvements après les moyennes ensemble pour former un fichier avec un nom/label (unique parmi tous les mouvements qui sera aussi le nom du fichier texte, sans espaces), les infos sur le mouvement avec erreur, sans erreur, les messages d'erreur, pour authentifier sa version. En plus de cela s'il n'existe pas encore un fichier maintenant une correspondance id → label&md5 qui sera créé (ou updaté avec une nouvelle entrée), qui enregistre une des lignes de la forme id → label&md5 (id un entier unique qui identifie le mouvement (générer comme id+1 l'id précédent (qui garantit l'unicité)) et label une chaîne de caractères en alphanum, md5 en alphanum également qui est le checksum md5 du fichier de mouvement). Input : faire plusieurs fois un mouvement donné devant la Kinect, les enregistrer, puis ensuite en récupérer des morceaux avant d'en faire la moyenne, faire ceci pour un mouvement avec quelques erreurs (en réutilisant les tests du module synthèse 3D), créer le fichier de mouvement Output : fichier de mouvement constitué comme souhaité

3.Kinect allégée + Classification + Synthèse 3D allégée Cad : Ajout en plus de ce qui précède d'une méthode permettant à partir d'un id de mouvement donné de trouver le mouvement correspondant dans le fichier précédemment décrit, qui retourne un message d'erreur si l'id n'existe pas, s'il existe permet de lire le fichier de mouvement correspondant (erreur s'il n'existe pas...), et de le décoder pour en stocker les informations dans un objet utilisable par le module

classification. En plus de cela on ajoutera l'affichage des messages correspondant aux erreurs (ou au mouvement bien effectué) dans les fonctionnalités de la fonction retournant le résultat de la classification. Input : mouvements prédéterminer devant la Kinect (pris dans les tests du module classification ou/et synthèse 3D...) Output : vérifier que le module de classification retourne les erreurs souhaitées

Ayant réussi ces premiers tests, il fallait pouvoir transposer les traitements évoqués précédemment, et créer une structure de base de données communes au PC et au smartphone qui puisse se mettre à jour rapidement d'où l'introduction d'un fichier d'association indexroot qui associe à chaque classe de mouvements l'indice correspondant à l'id évoqué dans le module classification.

4. Communication Client-Serveur + Android + PC Cad : Ajouter l'implémentation du client-serveur sur Android + PC (issue du module communication client-serveur), écrire dans un fichier les données reçues du côté PC TEST1 Input : à partir de fichiers de mouvements enregistrés sur le smartphone, les envoyer au PC Output : vérifier qu'ils arrivent en état avec le fichier TEST2 A partir de des méthodes de classification pour communiquer avec les fichiers de mouvements, créer 2 méthodes (une côté Android, l'autre côté PC) permettant de vérifier que le PC a bien la même base de données que celle du Smartphone en terme de mouvement (envoie du fichier des associations id → label&md5 depuis le smartphone, puis vérification que chaque mouvement est dans la base de données locale du PC, si un mouvement est manquant une requête est envoyée au Smartphone qui enverra le mouvement, si un mouvement est en trop il est supprimé et l'entrée retirée du fichier des associations id → label&md5). Input : 3-4 fichiers mouvements, obtenus lors des précédents tests et d'un fichier associations id → label&md5 correspondant, faire le test sans mouvements sur le PC Output : (en printant les changements à effectuer) vérifier qu'à la fin de l'opération la base de données est bien stockée localement Input : refaire le test avec la base de données bien stockée côté PC Output : (en printant les changements à effectuer) vérifier qu'aucun changement n'est imprimé Input : refaire le test avec un mouvement en trop du côté PC Output : (en printant les changements à effectuer) vérifier qu'il est bien effacé

5. Communication Client-Serveur + Android + Kinect allégée Faire 2 méthodes (côté Android, et côté Kinect), qui permettent à la Kinect (avec méthode pour fournir les données d'images reçues depuis la Kinect) d'envoyer des images au format jpg à Android et à Android de les sauvegarder dans un dossier local sous l'id de la séance Input : mouvement devant la Kinect avec sauvegarde d'images de manière aléatoire enclenchée Output : vérifier que les images sont bien parvenues sur Android et qu'elles sont lisibles

Ce dernier pan de test vise à faire fonctionner notre prototype allégé dans les conditions de démo.

6. Kinect allégée + Classification + Communication Client-Serveur + Android Permettre lors du lancement de l'application Android de vérifier la base de données du côté du PC, que la Kinect est allumée (2 nouvelles méthodes), et de récupérer les données issues de classification (2 nouvelles méthodes, on récupère l'id de l'erreur/mouvement bien réalisé, et on envoie l'id du mouvement + l'id de l'erreur/mouvement bien réalisé à Android qui peut s'en servir pour son affichage (normalement déjà implémenté dans le module Android) Les données seront de la forme : idMov – id_du_mouvement //un id de mouvement est envoyé

idMovError → [id_du_mouvement, id_de_erreur] //un id de mouvement et d'erreur est envoyé

image → image_date // une image au format jpg est envoyé

movFile → fileName fileData_L1 fileDate_L2 etc. // un fichier de mouvement est envoyé

index → indexFileName indexData_L1 indexData_L2 etc. //un fichier d'association d'id → label&md5 est envoyé

kinectOn -> //requête pour savoir si la Kinect est en marche

kinectOn -> boolean //réponse pour signifier que la Kinect est en marche (true) ou non (false)

Qui permet ensuite d'identifier les données des 2 côtés en modifiant légèrement les méthodes précédentes. Input : on allume l'application, on lance une séance avec la Kinect Output : on vérifie que l'application répond à nos attentes

5.8.2. Bilan PAN3

La partie communication client-serveur a été correctement intégrée. L'essentiel du travail restant réside dans le module Android, la gamification, et la concrétisation de certaines propositions du module SES.

5.8.3. Réalisations en vue du PAN4

A l'issue du PAN3, il restait essentiellement du travail en Android pour faire une interface graphique qui mette en valeur le côté joueur de SportBot, notamment avec la synthèse vocale.

Après avoir étudié les retours des entretiens SES, nous avons retenu l'idée d'implémenter des modes qui régissent l'interaction avec un mode sans interaction non intrusif en somme selon nos interlocuteurs, et un mode interactif qui correspondait à notre plan initial. Dans le mode non interactif, il est par exemple possible de passer la partie QCM correspondant aux goûts. L'application aura juste accès au nom/prénom, l'âge et la taille de l'utilisateur afin que le coach puisse l'appeler et faire les traitements de classification qui nécessite sa taille pour évaluer la distance à la base de la distinction des mouvements.

De plus, avoir la possibilité d'avoir un coach sévère ou non semblait être important en faisant le bilan de nos entretiens. L'idée est donc d'intégrer des transitions entre les exercices qui feront intervenir la synthèse vocale avec des messages. La tonalité de ces messages dépendra de la sévérité qu'a configuré l'utilisateur pour le coach bot, au début de la séance.

Une autre idée que l'on a retenu est la mise en place d'un site pour la communauté SportBot où on pourra avoir les scores des joueurs obtenus au Mirror Game expliqué dans le module Android. De même, l'utilisateur pourra voir des nouvelles mises à jour de SportBot. De plus, l'utilisateur aura accès à la vidéo, poster et rapport du projet pour que celui-ci puisse participer lui aussi au projet.

Par ailleurs, on a aussi la possibilité de faire une pause pendant les séances pour mettre de la musique, ceci se faisant par un claquement de main détecté par la Kinect. On s'est assuré de ne pas prendre des mouvements qui font se joindre les 2 mains régulièrement comme par exemple la posture de la chaise que nous avons finalement enlevée de la base de données de fichiers

mouvements.

Les transitions pourront aussi intervenir quand l'utilisateur reçoit 2 à 3 fois le même message d'erreur, avec des highlights qui apparaissent issus du travail préalable en synthèse 3D.

Pour implémenter la récupération des likes Facebook, nous avons pensé intégrer la demande de permissions en QCM rapidement. L'utilisateur pourra avoir des rappels des musiques qu'il a likées, ou des noms d'artistes. Le coach fait alors des suggestions sur ce que peut faire l'utilisateur durant la séance ou en dehors afin qu'il se sente le plus épanoui possible.