Université de Caen - Normandie

Rapport de projet annuel Master 2
ImaLang

Reconnaissance de postures avec la Kinect

Étudiant :
Yoann Bellet

Tuteur :
Youssef Chahir

Novembre 2015 — Février 2016

Table des matières

Introduction

Ce rapport sert de référence concernant le déroulement du projet annuel de seconde année de master Informatique, option Traitement de l'Image et de la langue. Ce projet concerne le domaine de l'imagerie et la reconnaissance de formes. Le tuteur de ce projet est M. Youssef Chahir, responsable du Master ImaLang et professeur d'imagerie à l'Université de Caen Normandie. L'objectif principal de ce projet est de réaliser une application permettant d'utiliser la caméra de Microsoft, la Kinect, comme outil de capture de mouvements sur plusieurs types de situations et de postures (assis, debout, marche, etc.). Nous allons dans un premier temps faire une présentation du projet et de ses objectifs, présenter ensuite ce qui à été réalisé, puis au final présenter des techniques possibles d'apprentissage et de suivi d'individus.

Chapitre 1

Présentation du projet

Dans cette partie du rapport nous allons détailler la composition de la Kinect ainsi que les différents enjeux de la reconnaissance de posture, et donc du projet.

1.1 Description

La Microsoft Kinect, connue à l'origine sous le nom de code "Project Natal" est un périphérique destiné aux consoles de Microsoft (xbox360 et xbox one). Elle permet un contrôle de l'utilisateur sur la console, et ce, sans utiliser de manettes. Elle à été conçue par Microsoft en septembre 2008.
Le mot "Kinect" est issu des mots Anglais "kinectic" et "connect". La sortie Européenne à eu lieu le 10 novembre 2010. Les composants de la Kinect sont :

- deux caméras de profondeur 3D
- une caméra couleur (RGB)
- un microphone

Ce sont la présence des ces différents capteurs dans l'appareil ainsi que le prix bas qui font de la Kinect un outil particulièrement bien adapté pour l'imagerie.

KINECT™
for XBOX 360®

FIGURE 1.1 – Kinect pour xbox 360

Cette caméra permet donc une détection d'individus plutôt efficace, d'autant plus avec des solutions tel que le SDK [1] de Microsoft.

Nous pouvons néanmoins nous orienter vers des solutions libres et multi-plateforme, comme OpenNI [2].

1.2 Mise en contexte

La reconnaissance de la posture d'un individu avec la Kinect au travers d'un logiciel permet une multitude d'applications dans différents cas [1] :

— **La surveillance** qui peut consister au suivi d'une ou plusieurs personnes dans le but d'analyser leur comportement. Cette surveillance

1. Kit de développement permettant de faciliter le développement d'un logiciel sur une plateforme, comme ici pour la Kinect
2. Voir le chapitre 2

peut être faite dans des lieux publiques ou dans le cadre privé pour prévenir d'actes malveillants ou de personnes en danger.

— **Le contrôle** effectué par la personne est ici réalisé afin que celui-ci prenne le contrôle de l'application (et donc de son environnement) avec la gestuelle pour effectuer des actions (ouvrir une page web, déplacer une fenêtre avec un mouvement de la main...)

— **L' analyse** peut être faite sur sa posture d'un individu. Il est possible par exemple détecter la manière de se déplacer d'un individu, permettant ainsi de détecter une maladie, comme les maladies neuro-dégénératives (Parkinson entre autre), ou la nécessité de faire appel à un podologue pour endiguer une déformation de la voûte plantaire. Les application analytique nécessite encore plus de précision pour être efficace.

1.3 Objectifs du projet

Parmi les différents objectifs au sein du projet, nous avons :

1. La mise en place de l'environnement de développement sous Linux avec des logiciels Open Source pour utiliser la Microsoft Kinect ;
2. Mise en place d'une architecture avec une IHM simple ;
3. Suivi de la posture et reconnaissance de la posture.

Chapitre 2

Application pour la visualisation de la caméra de la Kinect

Cette partie s'articule autour de l'application réalisée avec une présentation des capteurs et logiciels pour les utiliser, ainsi que des langages utilisés.

2.1 Les langages et IDE

Le langage utilisé pour ce projet est le C++, ce langage permet un calcul rapide, cette rapidité est nécessaire dans le cadre du calcul d'images avec la notion de temps réel. Afin de faciliter le développement, nous allons utiliser l'API Qt, permet de créer des interfaces graphiques simple et de concevoir une architecture simple. De plus, Qt possède un très bon IDE [1] : *Qt Creator*. Le principal attrait de Qt est son coté multi-plateforme, permettant de gérer du code aussi bien sur Linux que Windows ou Mac.

De plus, pour afficher les deux caméras de la Kinect (caméra RGB et caméra profondeur), nous allons utiliser la librairie d'imagerie numérique en C++ standard : *OpenCV*. De plus, elle possède un Wrapper [2] avec la bibliothèque OpenNI.

1. Environnement de Développement
2. Convertit l'interface d'une classe en une autre attendue par le client de la librairie.

(a) Langage C++ (b) API Qt (c) librairie d'imagerie OpenCV

FIGURE 2.1 – Langage utilisé et API associées

Néanmoins, si l'on veut utiliser la Microsoft Kinect sur un ordinateur, il nous faut installer un ensemble de drivers et de bibliothèque... D'autant plus que nous ne pouvons pas utiliser le SDK de Microsoft si nous souhaitons avoir une application multi-plateforme... Si nous voulons pouvoir développer sans payer de licences, nous devons nous tourner vers des alternatives libres.

2.2 Les logiciels libre utilisés

Pour réaliser le projet, nous avons utilisé des alternatives libres au SDK de Microsoft, mais pour remplacer ce kit de développement, nous avons besoin de plusieurs logiciels :

— **OpenKinect** est un logiciel créer par une communauté de développeurs intéressés par le développement sur la Microsoft Kinect. En créant libfreenect, ils ont permis une utilisation et reconnaissance de la kinect sur toutes les plateformes (USB, lecture de flux vidéo...).

— **OpenNI**, pour Open Natural Interaction est un logiciel open source permettant la certification et l'amélioration de l'utilisation d'appareils

permettant d'avoir une interface directe avec l'utilisateur (comme la Kinect).

— **NITE** est un middleware [3] permet à OpenNI d'obtenir une vision 3D de la caméra profondeur de la Kinect, permettant le suivi d'utilisateur, le contrôle à base de la main...

OpenNI et NITE sont deux logiciels crée par la société *PrimeSense* , racheté par *Apple* en novembre 2013 afin de faire stopper son activité.

(a) OpenKinect (b) OpenNI (c) NITE

FIGURE 2.2 – Logiciels utilisés pour reconnaître et exploiter le potentiel de la Kinect

2.3 IHM et présentation de l'Application

La première difficulté est de faire fonctionner l'ensemble des logiciels libres ensemble, afin de faire fonctionner la Microsoft Kinect, sur un environnement **Linux** pour ma part.

Ensuite l'objectif était d'obtenir une interface simple et minimaliste permettant de visualiser les différents flux de la Microsoft Kinect.

3. Logiciel agissant comme un "pont" entre un système d'exploitation et une application, ici, OpenNI

FIGURE 2.3 – Interface minimaliste de l'application

Nous pouvons voir sur cette interface deux deux cadres dans la partie supérieure, devant contenir la caméra RGB de la Kinect, ainsi que la caméra profondeur de la Kinect, ainsi que le squelette du suivi d'une personne. En dessous nous avons un terminal ou devront s'afficher les différentes informations de la Kinect (ainsi que des erreurs possible), mais aussi nous voudrions générer un log des postures simples dans un premier temps sous la forme :
`posture : <sit|stand|lie>_<HH:mm:ss>`
Et enfin un curseur permettant de régler l'inclinaison de la Kinect.

Les deux cadres devront contenir des flux semblables à ceux présents sur la figure 2.4.

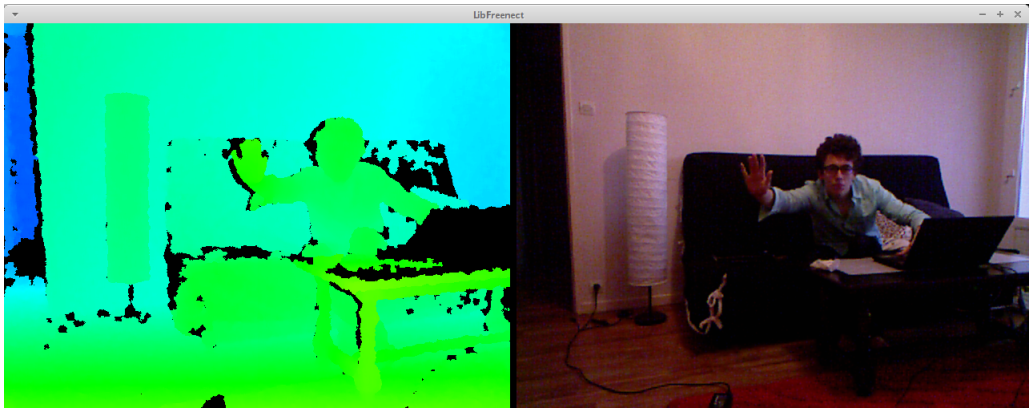

FIGURE 2.4 – Représentation de la caméras de profondeur (gauche) et couleur (droite)

La notion de profondeur dans les exemples d'OpenNI sont représentés par des couleurs plus ou moins froides à partir de la distance à la caméra.

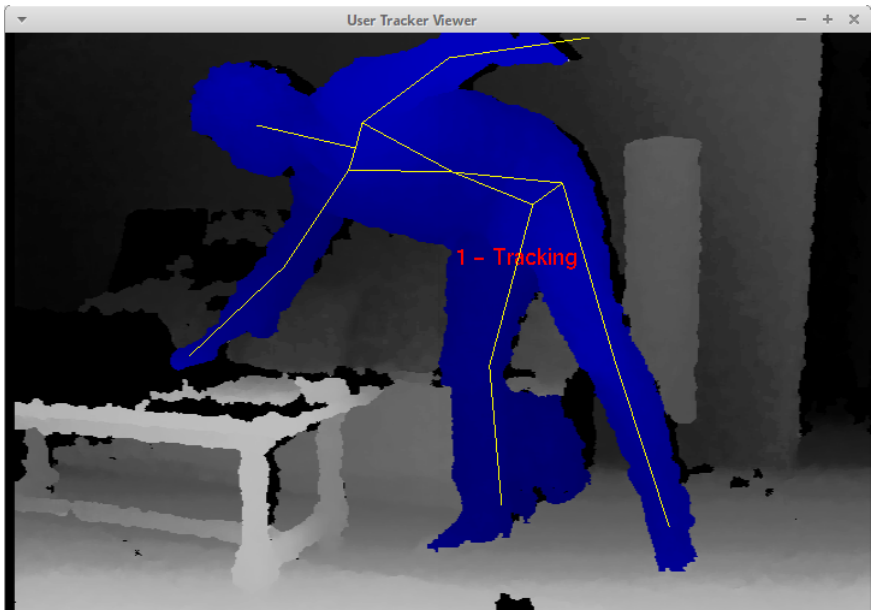

FIGURE 2.5 – Suivi d'un individu et représentation sous forme d'un squelette

L'autre exemple montre la manière dont laquelle OpenNI gère le suivi d'une personne et la représentation du squelette sous formes de points d'intérêt.

Chapitre 3

La reconnaissance de postures

L'intérêt du projet repose sur la reconnaissance de posture d'un individu, nous allons donc ici évoquer les intuitions que nous pouvons avoir concernant la partie de reconnaissance et d'apprentissage, génération des descripteurs...

3.1 Intuitions

La reconnaissance de mouvements nécessite au préalable la reconnaissance de l'individu ; c'est chose faite avec OpenNI qui, comme dans l'exemple précédant, permet d'obtenir une représentation du suivi de l'individu avec un ensemble de points d'intérêt. Mais de quelle manière pouvons nous les utiliser afin d'en déduire une posture ?

Il est possible de détecter les position en utilisant les positions des points d'intérêt de l'individu, et les angles entre certains de ces points [2].

Mais quels seraient les angles importants à utiliser ? Il est intéressant de regarder les angles générés par les points par rapport au référentiel terrestre :

- Épaule - Hanche - Genou
- Hanche - Genou - Cheville

Mais aussi d'utiliser les différentes distances entre :

- Tête - genou
- Épaule - poignet

Après extraction de ces données, il serait possible de créer des descripteurs afin d'en déduire la posture courante.

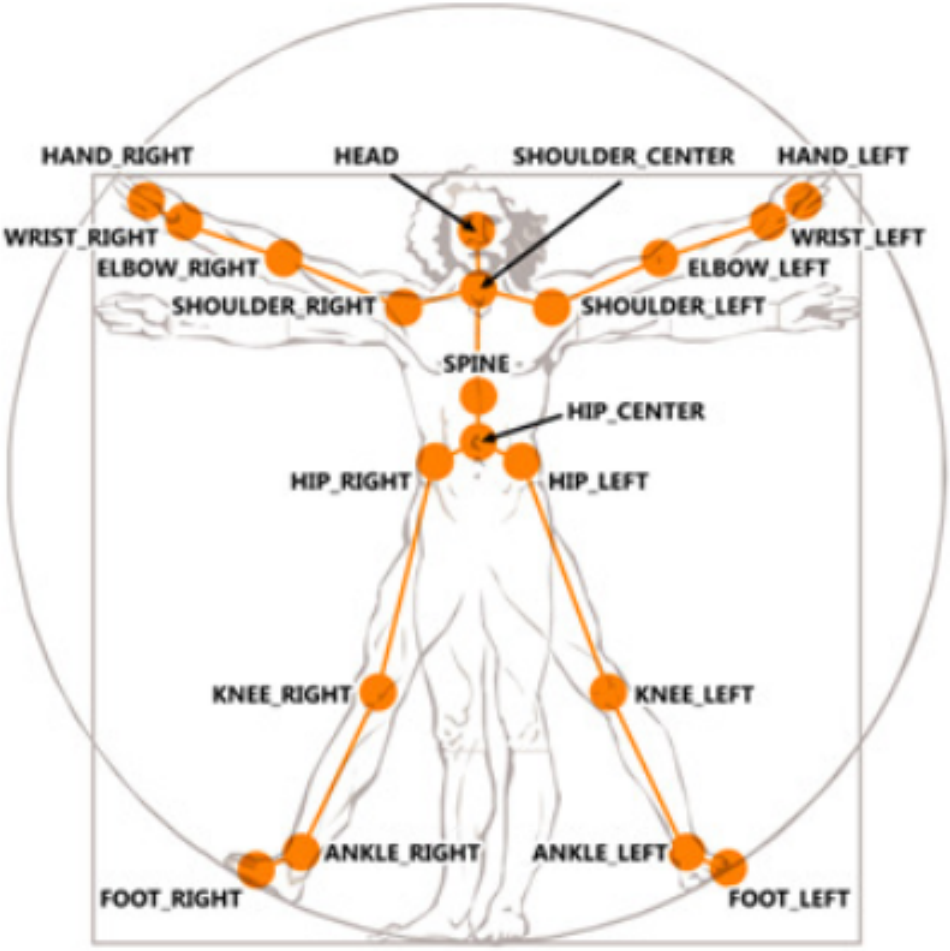

FIGURE 3.1 – Les points pouvant être repéré avec la Kinect

3.2 Apprentissage Automatique

Si l'on voulait utiliser les descripteurs récupérés à partir du suivi d'individu, il nous faudrait au préalable créer une base de connaissance afin de faire apprendre à notre application les différentes postures...

Ici intuitivement, on pourrait partir vers de l'apprentissage supervisé : On pourrait déterminer automatiquement une règle à partir de données d'apprentissage annotées au préalable (annoter un corpus d'individu que l'ont suit, puis annoter le descripteur de la posture à l'instant t. L'inconvénient est qu'il faudrait effectuer ce travail pour chaque type de posture que l'on voudrait reconnaître.

L'avantage est que la base de connaissance de l'application pourrait être en perpétuelle évolution, on pourrait par exemple demander l'annotation d'un expert médical afin de détecter des maladies agissant sur la posture. Néanmoins cette base demanderait beaucoup de travail.

Cette méthode pourrait être appliqué à des algorithmes simples, tel que l'algorithme des k-plus proches voisins, avec k le nombre de postures à reconnaître.

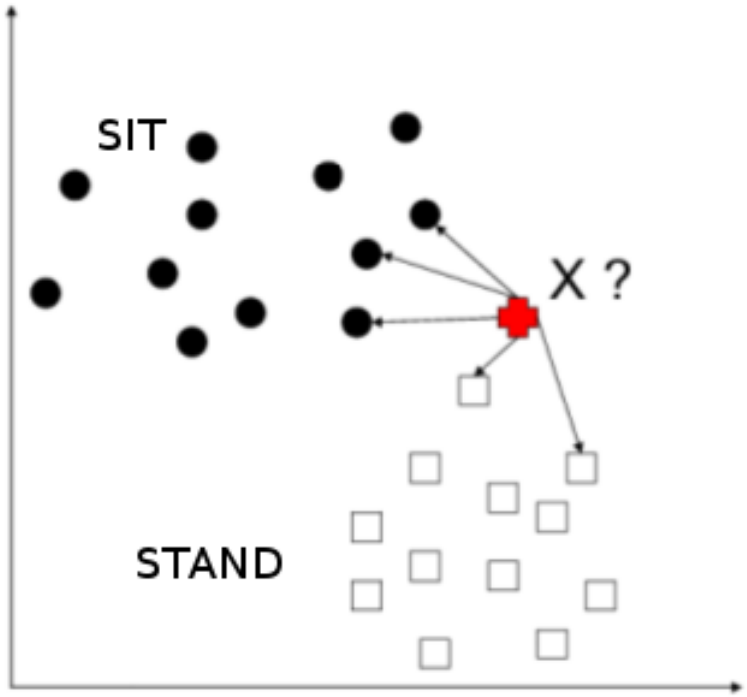

FIGURE 3.2 – Exemple d'apprentissage avec les Knn

Il serait aussi intéressant de tester nos descripteurs avec d'autres techniques d'apprentissage supervisé tel que les Support Vecteur Machine [3]. Les SVM sont une généralisation des classifieurs, pouvant etre facilement entrainer avec des bibliothèque comme *sklearn*.

Conclusion

Ce projet à un intérêt puisqu'il permet de reconnaître à partir d'une caméra bon marché tel que la Microsoft Kinect, la posture des individus passant devant ses caméra. Je n'ai malheureusement pas pu mener ce projet à bien, et ce pour plusieurs raisons :

- Une grande difficulté à utiliser les logiciels de **Prime Sense**, en effet, la société ayant été racheté par Apple, il n'y a que très peu de tutoriels ou d'explications d'utilisation utilisables aujourd'hui.
- Je n'ai pas réussi à lier les flux de la Kinect au sein de l'architecture de Qt.
- Un manque de temps, l'année de M2 étant très courte, avec de nombreux autres projets et partiels...

C'est pour quoi je n'ai fait qu'un travail de recherche théorique pour la troisième partie concernant l'apprentissage de posture.

Néanmoins, ce projet m'a semblé très intéressant (surtout d'un point de vue médical avec l'aide d'un expert dans les maladies neurodégénératives), mais sans utiliser les technologies libres, simplement le SDK de Kinect qui certes ne fonctionne pas bien sur d'autres système que Microsoft, mais possède bon nombre de tutoriels ainsi que des mises à jour fréquentes.

Bibliographie

[1] Bernard Boulay. *Reconnaissance de postures pour l'interprétation d'activité humaine*. PhD thesis, Université de Nice, 2007.

[2] Stéphanie Lopez Perside Gbehounou. Kinectition. `http://users.polytech.unice.fr/~gbehouno/contenu/Kinectition.php`, 2013.

[3] Thi-Thanh-Mai Nguyen Thi-Lan Le, Minh-Quoc Nguyen. Human posture recognition using human skeleton provided by kinect. *Computing, Management and Telecommunications*, pages 340 – 345, 21-24 Jan 2013.