Année 2013-2014

RAPPORT DE PROJET – EI4 AGI

# Interaction et visualisation de données médicales


*Projet réalisé par*

Maël Gadbois
Damien Guironnet
Clément Ménard
Yoann Neveu-Dérotrie

*Projet encadré par*

Jean-Baptiste Fasquel




ISTIA – Université d'Angers

# Remerciement

Nous tenons tout d'abord à remercier monsieur Jean-Baptiste Fasquel, notre tuteur, pour son aide, ses conseils et sa confiance tout au long du projet.

Nous souhaitons de même remercier M. José Braz, client et partenaire à l'université de Setúbal avec lequel nous avons pris plaisir à travailler et réaliser ce projet.

Enfin, nous remercions l'Istia pour le prêt du matériel ainsi que toutes les personnes avec qui nous avons pris contact et qui nous ont aidés.

# Contenu

# Présentation du projet

L'école est un lieu pour apprendre que ce soit de manière théorique mais aussi de manière pratique. Dans ce but à l'ISTIA, école d'ingénieur, chaque étudiant lors de sa quatrième et dernière année reçoit un projet pour ainsi mettre en pratique les compétences apprises durant ses heures de cours.

Notre groupe est composé de Maël Gadbois, Damien Guironnet, Clément Ménard, Yoann Neveu-Dérotrie. Sous la tutelle de Jean-Baptiste Fasquel, nous avons développé un projet en partenariat avec l'université de Lisbonne. Le but de celui-ci était de réaliser un programme permettant à un utilisateur quelconque de manipuler à distance, sur un écran, une image médicale dans un monde virtuel en 3D.

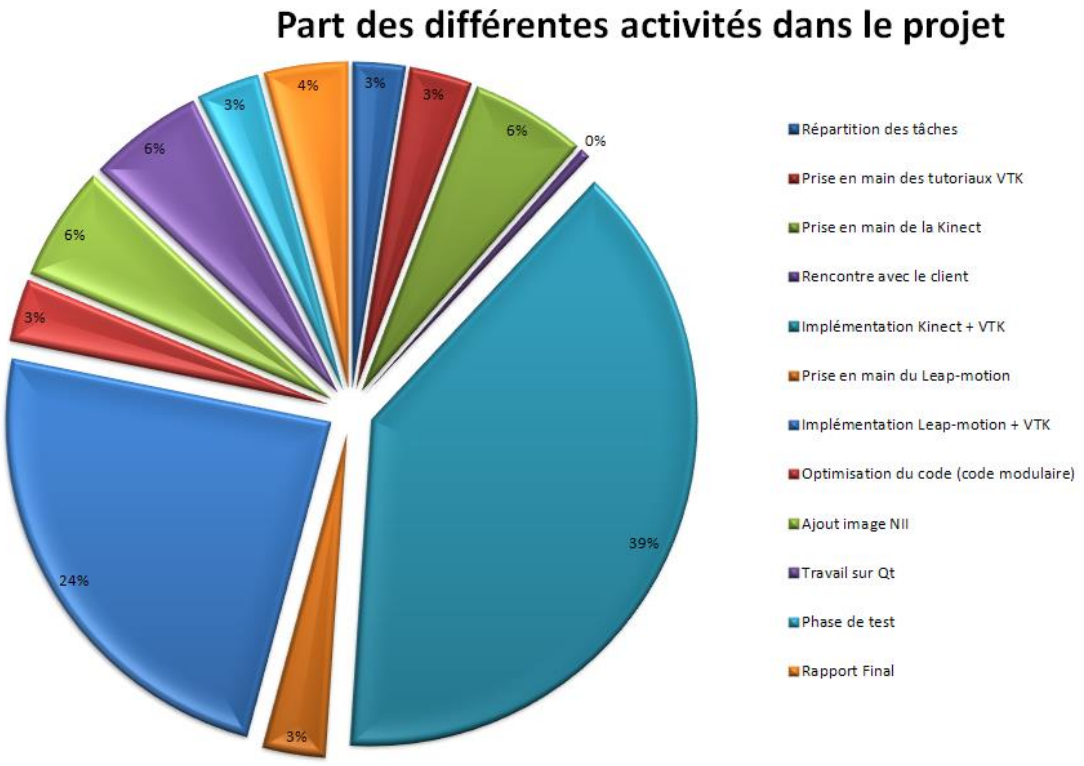
Ce projet a principalement but de pouvoir s'intégrer dans un environnement médical pour pouvoir aider les médecins lors d'opérations par exemple. En effet, en ce moment, ceux-ci dispose de nombreuses données mais ces dernières peuvent s'avérer parfois dur à utiliser. Nous avions pour premier objectif de modéliser les données médicales (radiographie, imagerie par résonance médicale IRM, etc). Le second était de pouvoir déplacer les volumes 3D préalablement crée une Kinect ou un Leap Motion. L'avantage principal de ces deux outils sont qu'ils peuvent s'utiliser sans contact, ce qui lors d'une opération permet au chirurgien de faciliter sa visualisation des données. De plus cela permet de limiter les risques d'infection.

# Gestion du projet

Pour pouvoir travailler efficacement pendant le projet, dès les premières séances, nous avons décidé de diviser le projet en plusieurs parties et étapes. Ainsi nous avons formé deux groupes, l'un s'occupant de la partie Kinect, l'autre du Leap. Nos premières séances ont ensuite été consacré a la prise en main de l'environnement python mais surtout VTK à l'aide des scripts fournis par notre tuteur.

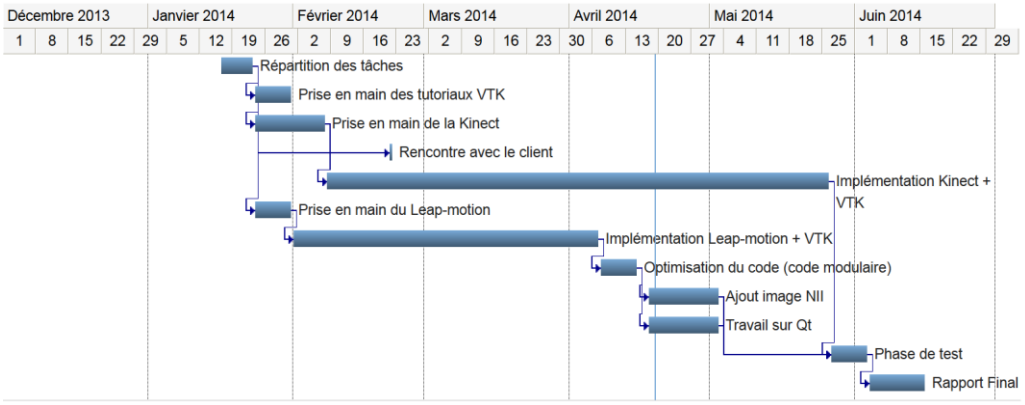
Ensuite, nous avons récupéré les équipements (Kinect et Leap Motion) pour que chaque groupe puisse se mettre au travail. Les installations et la prise en main du matériel a parfois était compliqué ce qui permis au groupe "Leap" d'avancer plus sur le projet et donc de proposer plus de fonctionnalité comme cela sera expliqué plus tard dans ce rapport.

Sur les figures ci-après, vous pourrez retrouver la part en temps des différentes parties du projet ainsi qu'un diagramme gantt récapitulant l'ensemble de celui-ci.




Le projet étant en partenariat avec une université portugaise, deux après-midi ont été consacrés à la rencontre d'un de leur responsable (M. José Braz). Cela nous a permis de mettre en place un cahier des charges visant à connaître ce qu'il attendait de nous mais aussi de voir notre avancement. Pour faciliter les échanges, nous avons travaillé sur un système de stockage en ligne leur permettant de voir le développement du projet mais aussi de pouvoir tester les

différents scripts que nous avons pu écrire. De plus une correspondance mail s'est établie pour répondre aux différents problèmes et questions qui ont pu apparaître. Par ailleurs nous tenions régulièrement informé M. Fasquel, notre tuteur et allions le voir lorsque nous avions des problèmes pour tenter de les résoudre.



# Visualisation des données médicales

## Dicom

La première image que nous devions manipuler était de format DICOM. Le format Dicom (Digital Imaging and COmmunications in Medicine), est une norme standard pour la gestion informatique des données issues de l'imagerie médicale.

VTK (Visualization ToolKit) est une bibliothèque libre pour la visualisation de donnée et le traitement d'image. Le format Dicom peut être gérer avec VTK, on obtient donc rapidement une image interprétable par VTK.

```
dicomReader = vtk.vtkDICOMImageReader()
dicomReader.SetDirectoryName('data')
dicomReader.Update()
vtkImage = dicomReader.GetOutput()
```

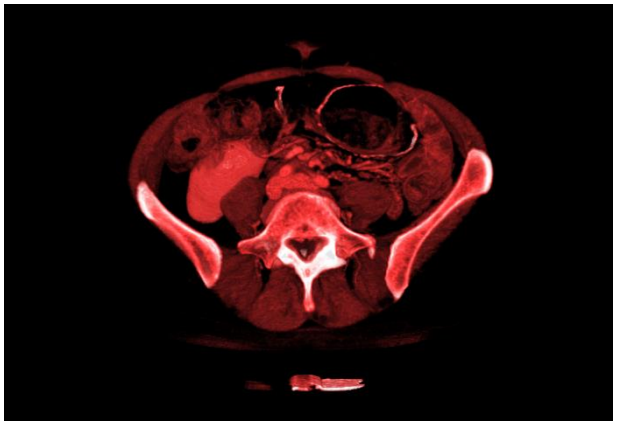


*Image Dicom*

## NII

La deuxième image est au format "NIfTI-1 Data Format" (.nii). Ce format n'est pas géré par VTK. Il faut donc importer une autre librairie dénommé Nibabel. Cette dernière va permettre de récupérer les données de l'image. Une fois les données récupérer, il faut insérer ses élément dans une image vide créée par VTK.

```
Image = nibabel.load('im1.nii')   #chargement de l'mage
data = Image.get_data()
vtkImage = vtkImageImport() # on crée une image vide
data_string = data.tostring() #on récupère les informations de l'image sous la forme d'une chaine
vtkImage.CopyImportVoidPointer(data_string, len(data_string))  #on importe les informations de l'image
                                                                nibabel dans l'image VTk
```

Une fonction de rendu volumique permet de donner un effet en trois dimensions. Cette fonction a pour paramètre, l'image, l'intensité et l'opacité que nous souhaitons donner à cette image.

Une fois que le volume est créé (image en trois dimension), elle est ajoutée à une fenêtre VTK avec la commande "window.AddRenderer". La fenètre peut ensuite être ajouter avec "windows.render".

Il ne faut pas oublier de créer un interactor pour pouvoir manipuler l'image.

```
interactor = vtk.vtkRenderWindowInteractor()
interactor.SetRenderWindow(window)
```

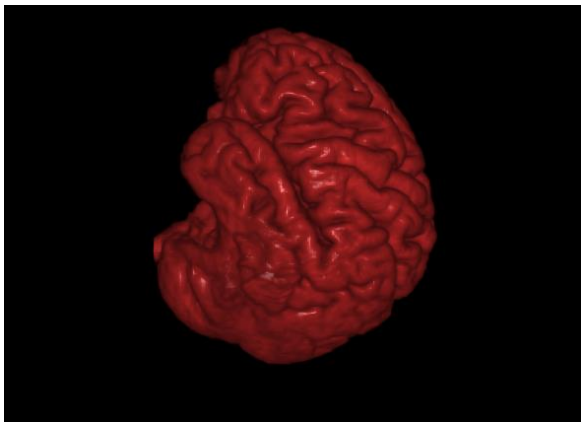
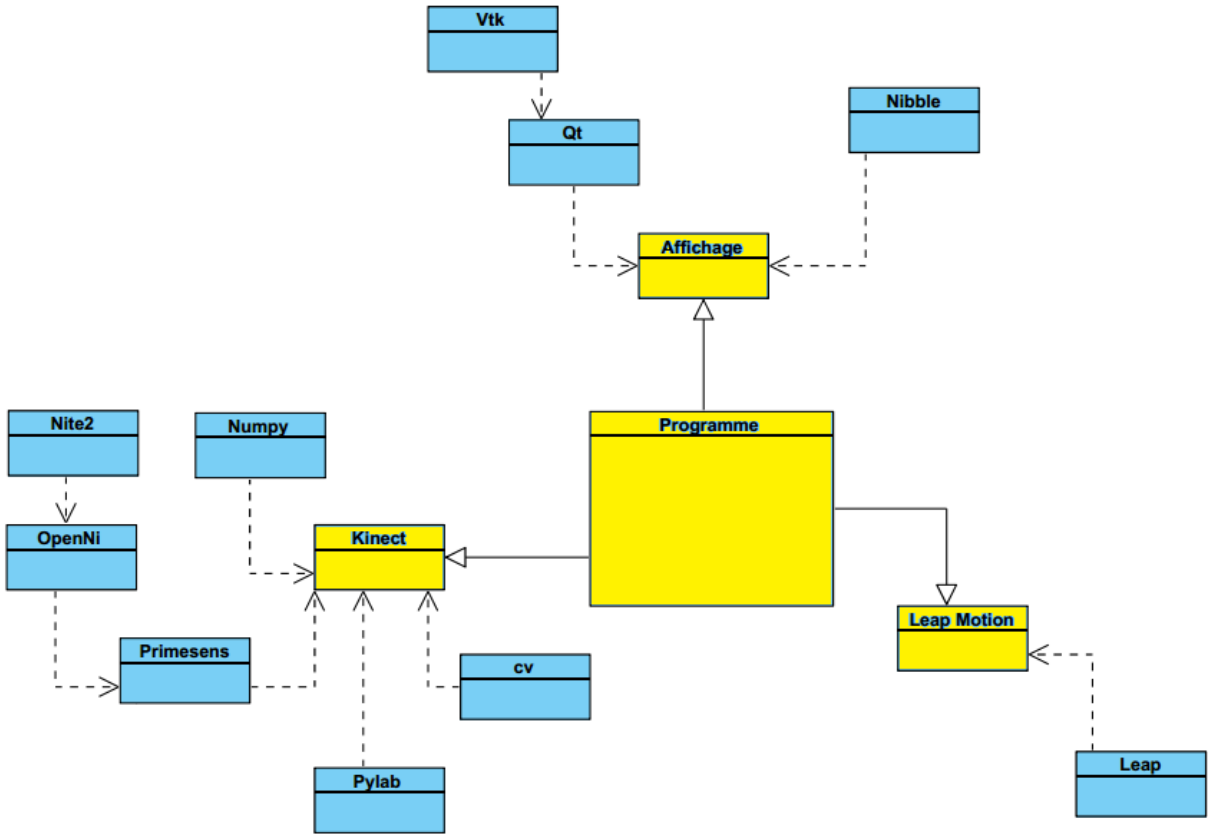


*Image Nii*

# Schémas




Dans cette partie, nous avons décidé d'expliquer un peu plus en détail, l'architecture de notre programme. Celui-ci comporte trois grandes parties nécessaire à son fonctionnement (en jaune sur le schéma : l'affichage, la partie Kinect et enfin la partie Leap Motion.

Chacune de ses branches a besoin d'importer des librairies propres pour pouvoir fonctionner (comme cela est représenté). Pour le fonctionnement du projet, il est donc nécessaire d'installer chacune de ses librairies et parfois dans un certain ordre (cf. IViMeDa_InstalationGuide.pdf, guide d'installation que nous avons réalisé en collaboration avec José Braz).

# LEAP

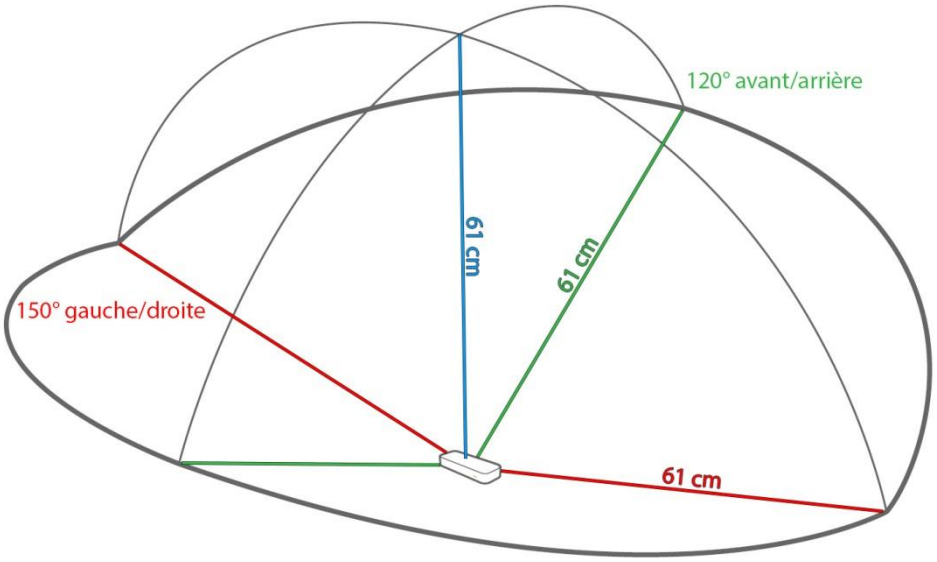
## Présentation de le leap

Le Leap Motion Controller développé par Leap Motion Inc. est une des dernières nouveautés en matières d'interaction hommes machines. Ce périphérique peut être connecté en USB et permet une fois placé sur une surface plane de récupérer des informations sur les mains d'un utilisateur. Il est très peu encombrant (une dizaine de centimètres de long) et se trouve même intégré à certains ordinateurs portable et claviers.

Ce dispositif a été présenté à sa sortie comme étant très précis. L'appareil est en effet environ deux cent fois plus précis que la première Kinect mais détecte uniquement les mains.

Les différentes informations que nous pouvons obtenir comprennent la position des mains, leurs inclinaisons, la courbure de la main, le nombre de doigts apparent, si le poing est fermé…

D'un point de vue technique le Leap Motion Controller est équipé de trois LED et deux caméras infrarouges filmant vers le haut en stéréoscopie. Ceci va créer un champ d'un peu moins d'un mètre de diamètre dans lequel seront détectés les mains et doigts de l'utilisateur.




*Aire de détection du Leap Motion Controller*

Contrairement à une souris ou un écran tactile, le Leap Motion Controller permet une gestuelle en trois dimensions d'où son intérêt pour la manipulation d'image en trois dimension notamment médicale.

## Programmer avec la LEAP

La première étape pour se servir du Leap Motion Controller est l'installation du périphérique et de télécharger le SDK (Software Development Kit) qui permet d'utiliser les librairies de Leap Motion Inc.

Une liste des différentes classes les plus utilisé de l'API Leap Motion :

Controller : L'interface entre l'application et le Leap Motion
Listener : Gère les évènements récupérer par le Leap
Frame : Contient les informations des mains et doigts détectés par le Leap Motion
Hand : Contient les informations des mains détectées par le Leap Motion
Finger : Contient les informations des doigts détectées par le Leap Motion
Vector : vecteur en trois dimensions suivant l'orientation de la main
Gesture : Permet de gérer différents mouvement pré-enregistré.

```
listener = SampleListener()
controller = Leap.Controller()
controller.add_listener(listener)
```

Pour faire un programme qui fonctionne avec la Leap Motion il faut ensuite initialiser le listener et le controller pour ainsi pouvoir récupérer les informations du Leap Motion Controller et qu'elles agissent dans le programme :

```
hand = frame.hands[0]  #On récupère les informations concernant la première main trackée
hand.direction              #Récupère la direction de la main (trois dimensions)
hand.palm_normal       #Récupère la normal à la main (trois dimensions)
hand.fingers     #Récupère les informations correspondant au doigt de la première main trackée
```

On initialise ensuite le frame : frame = controller.frame()

On peut ensuite à l'aide du frame récupérer les informations correspondant aux mains (frame.hands) et aux doigts de l'utilisateur (frame.fingers).

Pour le projet il nous fallait détecter différents mouvement qui pouvaient agir sur une image en trois dimensions. Nous avons donc essayé les fonctions prédéfinies dans la classe Gesture qui permettaient de récupérer des gestes simples. Il faut tout d'abord activer la reconnaissance des gestes puis avec une simple condition "if" décider de l'action à effectuer en fonction du mouvement.

```
controller.enable_gesture(Leap.Gesture.TYPE_CIRCLE) #faire un cercle avec un doigt
controller.enable_gesture(Leap.Gesture.TYPE_SWIPE) #faire un mouvement de balayage
if gesture.type == Leap.Gesture.TYPE_CIRCLE #La gesture Cercle a été détecté
```

Nous avons décidé de ne pas utiliser la classe Gesture pour avoir plus de liberté dans le choix des mouvements. Pour une manipulation simple de l'image nous avons choisis que notre programme reproduirait sur l'image la position et la direction de la main. Nous avons donc deux fonctions principales qui sont la rotation et la translation.

La rotation récupère la direction et la normal à la main. On calcule ensuite le pitch, le roll et le yaw de la main. On a plus qu'à effectuer une rotation de l'image en fonction de ces trois paramètres. Nous avons inclus un coefficient d'amplification qui permet d'éviter des mouvements trop amples du poignet lors de la manipulation de l'image.

La translation récupère la position de la main en trois dimensions dans l'espace (hauteur, mouvement dans la largeur et dans la profondeur). Il va ensuite changer l'échelle s'il y a un mouvement dans la profondeur (zoom, dézoom), effectuer des translations pour un changement de hauteur et un mouvement dans la largeur.

Pour éviter que la rotation et la translation se fasse en même temps nous avons décidé que la translation ne se ferait que si le poing est fermé (le Leap Motion ne détecte pas de doigt). Le poing fermé correspond donc à une "saisie" de l'objet. La rotation se fait donc quand la main est ouverte.

Pour fonctionner notre programme utilise un timer et l'interactor de la fenêtre VTK. Chaque 50 millisecondes un event "timer_event" appelle les fonctions "rotation" et "translation" avec l'interactor de VTK ce qui permet la manipulation de l'image.

## Problèmes rencontrés

Le Leap Motion Controller est relativement facile à installer. De nombreux exemples sont fournis sur le site internet (notamment en python), et un exemple est donné avec le SDK. La prise en main est donc plutôt facile. Les problèmes sont survenus lorsque nous avons voulu connecter notre programme qui se servait du Leap Motion Controller avec le VTK et l'affichage de l'image médicale.

Nous avons tout d'abord essayé de créer plusieurs threads fonctionnant en parallèle. Le premier thread devait gérer l'affichage de l'image et le deuxième récupérerait les mouvements des mains. Les threads nous ont posé quelques problèmes que nous avons contournés en utilisant les timer event.

# KINECT

## Présentation de la Kinect

La Kinect a été conçue par l'entreprise Microsoft dans le but de fonctionner avec la console de jeux Xbox 360. Son objectif était de permettre aux utilisateurs de la Xbox de jouer à certain jeux-vidéos sans aucun autre périphérique autre que leur propre corps. En effet, la Kinect est une caméra qui permet de capturer des images et ainsi de repérer des mouvements par le traitement d'images.


KINECT for XBOX 360.

La Kinect capture une image de l'ensemble du corps humain, ce qui peut s'avérer très pratique par rapport au type d'application que l'on souhaite faire avec.

## Cahier des charges

Notre travail a été, dans un premier temps, de récupérer à l'aide de la Kinect les images de l'utilisateur.

OpenNI analysait ces images afin de repérer les deux mains de la personne en action et comparait les images reçues entre elles afin de trouver les mouvements et gestes effectués par l'utilisateur. Puis nous devions assigner ces gestes à des mouvements sur l'image visualisée.

Voici quelques exemples des gestes qu'on souhaitait implémenter au cours du projet :

| Main | Mouvement de la main | Mouvement de l'image |
|---|---|---|
| **Main droite** | Translation Gauche -> Droite | Rotation horaire autour de Y |
| | Translation Droite -> Gauche | Rotation anti-horaire autour de Y |
| | Translation haut -> bas | Rotation horaire autour de X |
| | Translation bas -> haut | Rotation anti-horaire autour de X |
| | Translation avant -> arrière | Zoom in (translation de l'image vers +Z) |
| | Translation arrière -> avant | Zoom out (translation de l'image vers -Z) |
| **Main gauche** | Translation Gauche -> Droite | Translation Gauche -> Droite |
| | Translation Droite -> Gauche | Translation Droite -> Gauche |

Explications :

Ce fonctionnement avec deux mains est basé sur celui des analogues d'une manette de la console de jeux Playstation : L'analogue de droite change l'orientation de la caméra et celui de gauche dirige le personnage ou le véhicule.

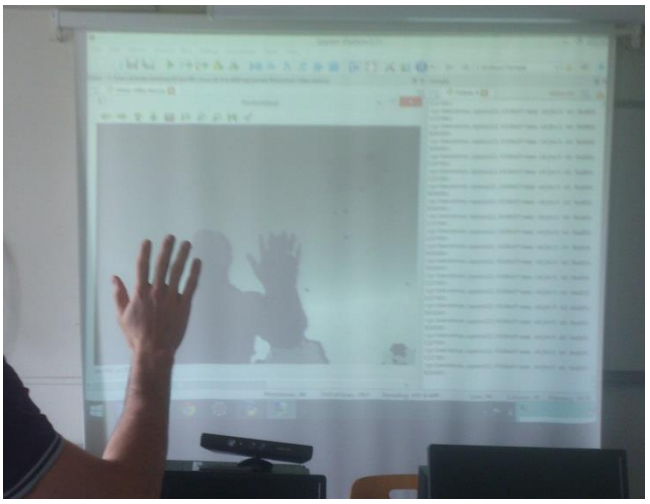

*Manette Playstation*

Nous nous sommes donc inspirés de la manette Playstation dans le sens où chaque analog gère une "catégorie" de mouvements différente :

- La main gauche gère les déplacements en translation de l'image
- La main droite gère la rotation de l'image ainsi que son zoom

## Notre réalisation

Nous avons commencé par afficher le retour vidéo de la Kinect.

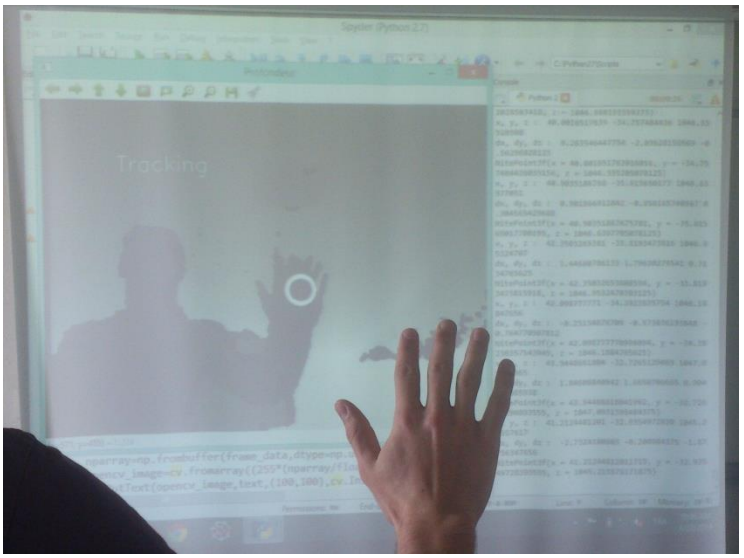


*Retour vidéo de la Kinect*

Nous avons ensuite repéré lorsque l'utilisateur approchait sa main devant la Kinect. À ce moment-là, nous commencions le "tracking" de la main.

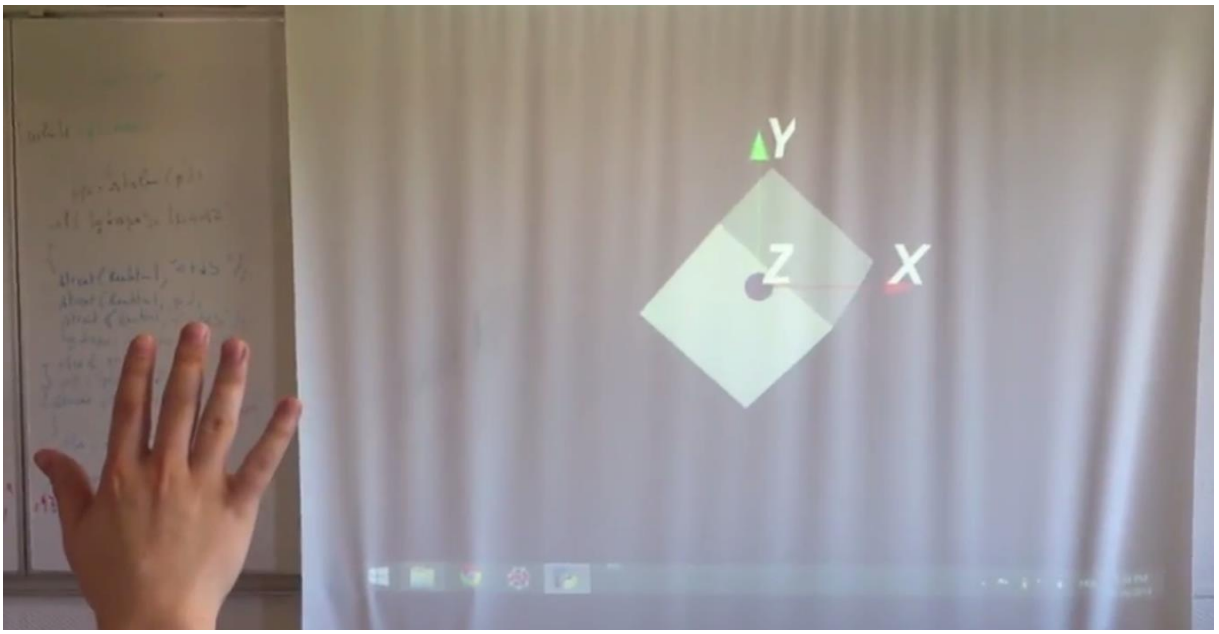
Le "tracking" correspond à la récupération des coordonnées de la main sur deux images différentes, puis la comparaison de ces coordonnées, ce qui permet de "suivre" la main image par image et ainsi d'obtenir les mouvements opérés par l'utilisateur.

Nous avons donc pu réaliser l'affichage de la main traquée avec OpenCV : retour vidéo transmis par la Kinect et ajout d'un symbole (ici un cercle) afin de marquer visuellement le "tracking" de la main.

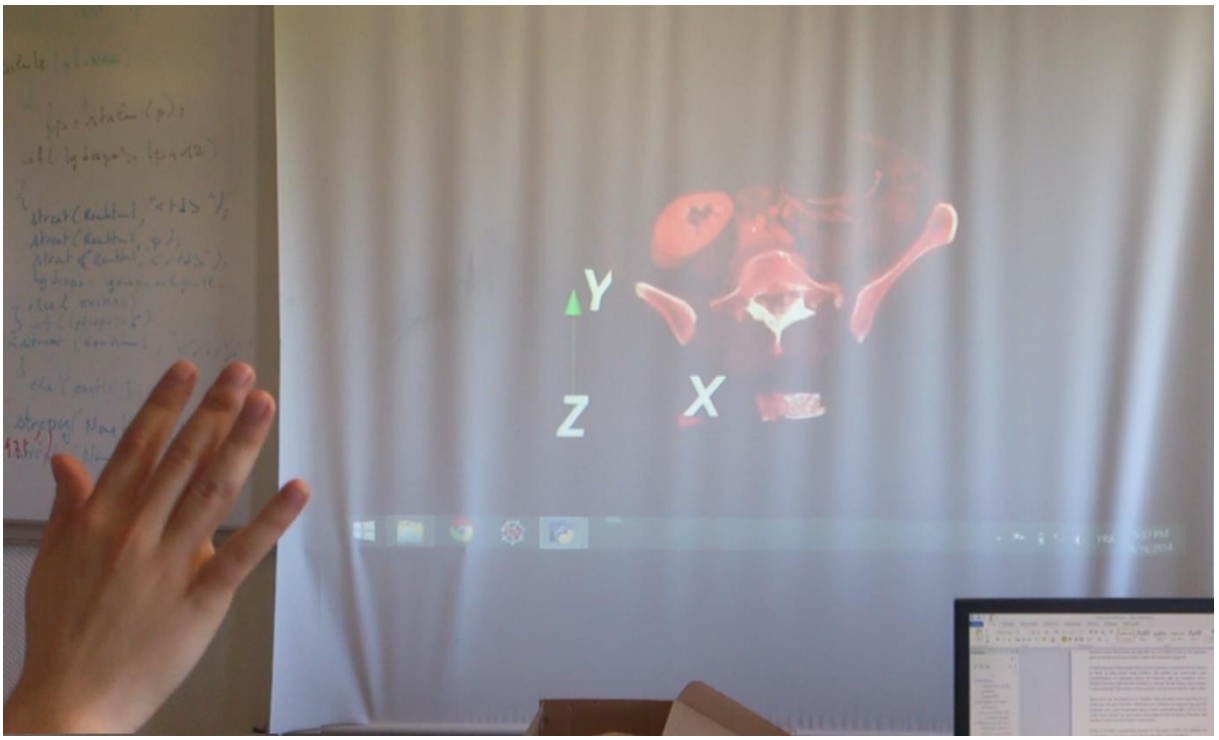



*Tracking sur une main*

Nous avons ensuite appliqué des mouvements de la main de l'utilisateur à un cube affiché avec VTK.



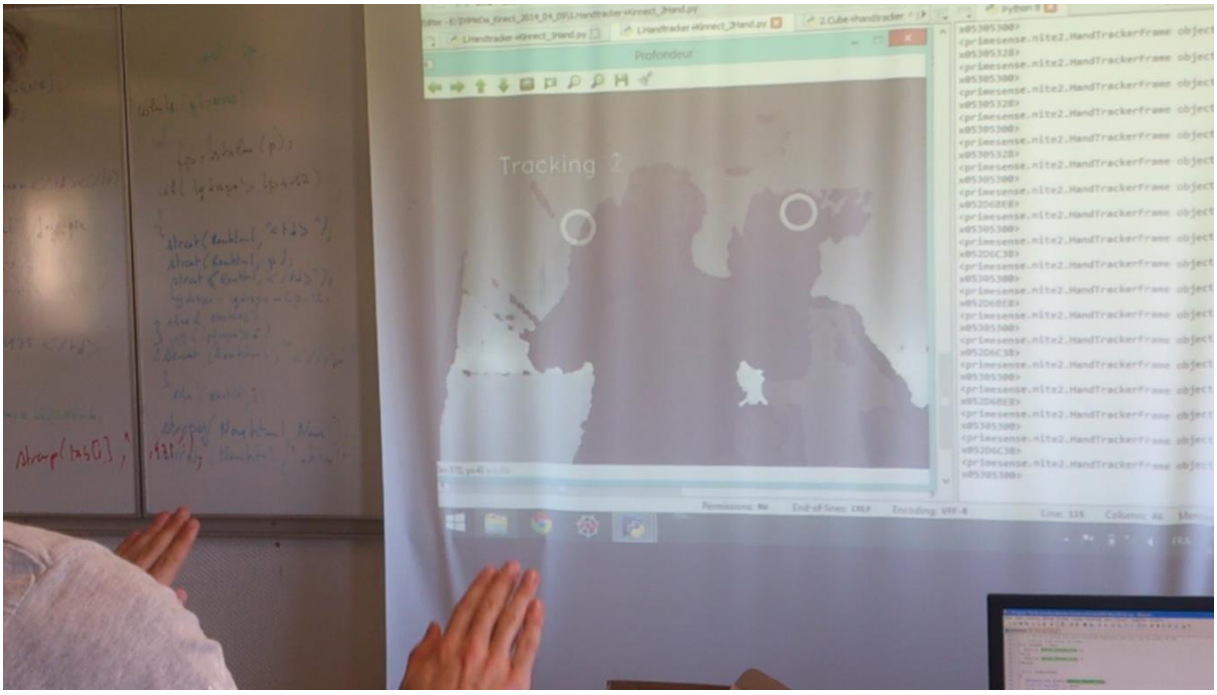

Après avoir, retransmis certains mouvements de la main de l'utilisateur à un cube, nous avons pu faire la même chose avec l'image médicale du bassin.




Contrairement au Leap Motion Controller, il n'est pas possible de reconnaître, avec la Kinect, les gestes de rotation de la main sur elle-même. Nous ne pouvions donc pas réaliser tous les mouvements souhaités avec seulement avec une seule main.

Par conséquent, parallèlement à la mise en mouvement du cone et de l'image médicale, nous avons cherché à traquer deux mains simultanément afin d'obtenir cette plus large palette de mouvements possibles.

Nous avons réussi à faire cela, mais malheureusement, par manque de temps, nous n'avons pu implémenter les mouvements "deux mains" au cône et à l'image médicale.




*HandTracking sur deux mains*

## Problèmes rencontrés

Nous avons passé un grand nombre d'heures à l'installation de la Kinect. En effet, nous avons eu du mal à trouver les différents éléments (OpenNI, Nite 2, "bindings" python, etc…). Les différentes aides, explications que nous avons trouvées sur des forums et sites internet ne se recroisaient pas tout le temps.

De plus, les différents exemples de projets que nous avons pu trouver sur internet étaient en C++.

Nous avons finalement réussi à installer la Kinect. Cependant, c'est dommage que nous ayons passé la plupart des heures de projets à installer la Kinect et à trouver les bindings python et que nous n'ayons vraiment commencé à créer un programme fonctionnel seulement lors des dernières semaines du projet car nous aurions pu alors aller bien plus loin !

# Différences entre les deux services

Au cours de ce projet, il nous a été demandé d'utiliser deux périphériques afin de programmer la manipulation des images médicales.

Malgré le fait qu'ils ai le même objectif de fonctionnement (capture de mouvements), la programmation sur chacun d'eux à été particulièrement différente sur quelques points :

- Installation des périphériques :

Pour la Leap motion, l'installation a consisté à installer les drivers, ainsi que le SDK du périphérique.

Pour la Kinect, la procédure d'installation a été un peu plus laborieuse : il fallut, tout d'abord, installer le SDK pour la Kinect, OpenNI, puis Nite2 et enfin faire les "bindings" python.

- Documentation de programmation :

La documentation de l'API pour la Kinect ainsi que les exemples n'étaient malheureusement disponibles qu'en C++. Cependant, grâce à notre tuteur, nous avons pu obtenir la correspondance des fonctions et classes en python, ce qui nous a été d'une grande aide.

Quant au Leap Motion, la documentation en python a été bien plus facile à trouver et surtout, bien plus fournie !

- Détection de la main :

La Kinect est moins précise que le Leap Motion pour la détection de la main, elle ne peut, par exemple, détecter lorsque la main effectue une rotation sur elle-même. Cependant, la Kinect permet de détecter de plus loin : plusieurs mètres pour la Kinect et environ 1 mètre pour le Leap motion.

Enfin, les deux périphériques peuvent détecter deux mains en même temps. Le plus de la Kinect est qu'elle peut aussi détecter le corps entier de l'utilisateur.

# Améliorations possibles

Nous n'avons eu le temps d'achever à 100% notre projet. Des améliorations sont envisageables :

## Point de vue du Leap Motion

Une première amélioration pourrait être au niveau de la fluidité. On peut remarquer que lorsqu'on passe du mode rotation au mode translation (changement de posture de la main), l'image peut avoir tendance à faire des mouvements non souhaités.

Une deuxième amélioration aurait été de créer un geste qui permette de réinitialiser l'image dans sa position d'origine pour recommencer la manipulation de zéro. On pourrait se servir de ce geste dans les cas où l'on a du mal à recadrer ou centré l'image.

## Point de vue de la Kinect

Il faudrait encore ajouter à l'image médicale les mouvements de la deuxième main de l'utilisateur. Cependant on pourrait dire que ce serait seulement un "petit plus" car les 3 rotations que nous avons déjà réalisées avec une seule main associées au Zoom/Dézoom, nous permettent d'ores et déjà de pouvoir visualiser l'image médicale sous tous les angles souhaitables.

## Point de vue global

Nous aurions pu ajouter des menus permettant, par exemple, de charger une nouvelle image, de modifier l'interface (couleurs, etc…), de changer l'apparence de l'image elle-même (transparence, couleurs, etc…).

Il serait aussi, pourquoi pas, envisageable d'ajouter une option qui permettrait de charger deux images côte à côte et ainsi, l'utilisateur pourrait comparer les deux images en les bougeant soit en simultanée, soit une par une en les sélectionnant chacune à son tour.

## QT

Après avoir affiché l'image dans un environnement vtk, nous voulions créer une fenêtre QT qui contiendrait l'image et permettrait de rajouter des menus de manière à créer une interface. L'interface nous aurait permis de choisir l'image à ouvrir ou de changer les paramètres de l'image que l'on veut manipuler. On peut par exemple changer l'opacité de manière à faire apparaître de nouveaux éléments sur notre image.

Cette partie du projet n'a pas abouti. Une fois l'image présente dans la fenêtre QT nous ne pouvions plus la manipuler. Pour insérer l'image dans la fenêtre QT, il faut créer un widget dans lequel on ajoute l'image médicale modélisé par VTK.

# Apports personnels

Au fur et à mesure du projet, nous avons pu découvrir et approfondir beaucoup de notions déjà vu en cours ou non. En effet, réaliser tout le projet en python, nous a permis d'aller plus en profondeur sur ce langage, donc de mieux l'appréhender et de voir son utilisation sur un cas concret. L'un des avantages qui nous a le plus marqué est la diversité et la multitude de librairies qui peuvent être utilisé par python. Pour notre part, nous avons seulement utilisé celle permettant de réaliser du traitement d'image et de la visualisation d'espace et d'objets en 3D, nous faisant ainsi revoir quelques notions vu en cours de réalité virtuelle et de vision industrielle.

Le projet a aussi pour nous tous été un moyen de travailler pour la première fois sur la Kinect et le Leap Motion, ce qui nous a fait découvrir le monde de la recherche. En effet, nous avons dû créer une application de nous-même en n'ayant aucun énoncé mais seulement un cahier des charges et l'aide de notre tuteur. Cela est une bonne chose puisque, cela nous permet de mettre en pratique toutes nos connaissances vu les années précédentes, de rechercher celle qui nous manque, tout en restant autonome.

Étant un projet de recherche, nous avons dû rechercher. C'était en quelque sorte une première puisqu'il a fallu tout d'abord lire les documents déjà existants, les adapter, trouver de nouveaux services et idées à implémenter, tout en faisant notre propre documentation, pour que d'autres personnes puissent se servir de nos travaux. Cela est un plus et une expérience que nous ne pouvons avoir en cours.

Ce projet a de plus renforcé la volonté de deux d'entre nous de vouloir choisir l'option IHM-RV en 5ième année.

En conclusion, ce projet a été pour nous un moyen de se projeter dans le futur en tant "qu'ingénieur". Travailler en équipe. Être autonome. Développer un produit pour un client. Être en contact avec un autre organisme.

# Bibliographie / Webographie

- https://www.leapmotion.com/

- http://www.clubic.com/technologies-d-avenir/article-575170-1-leap-motion-test.html

-https://developer.leapmotion.com/documentation/python/devguide/Sample_Tutorial.html

-http://www.vtk.org/doc/release/4.0/html/

- http://nipy.org/nibabel/

- http://www.microsoft.com/en-us/kinectforwindows/

-http://www.primesense.com/wp-content/uploads/2013/04/PrimeSense_NiTE2API_ProgTutorialGuide_C++Samples_docver0.2.pdf

-http://overconsulting.net/posts/installation-d-une-kinect-sous-ubuntu-13-10-openni2-freenect-nite2

- http://www.vtk.org/doc/nightly/html/index.html

# Interaction et visualisation de données

Projet réalisé par : Maël Gadbois, Damien Guironnet, Clément Ménard, Yoann Neveu-Dérotrie
Projet encadré par : Jean-Baptiste Fasquel

## Abstract:

Actual systems of medical data's visualization do not permit easy manipulation for medics. The project goal was to simplify the interactions and make it touchless. The original idea was to develop a new interaction method with electronic images (MRI, X-Rays, etc…). Some other projects about this topic were already done but not with the all new hardware available for this project. An infrared hand tracker named Leap Motion was used to have the position, orientation and number of hands and fingers. With those data, different source codes were written in Python to analyze movements and "teach" the computer human interactions. After this part was done, the only last thing to program was to link the movements of hands, to the movements of virtual objects on the computer. The objects are given by a medical researcher and represent brain, stomach or any other organ. The hardware used (Leap Motion) is a lot more precise than other hardware used before. The results are for the moment very positive and control 3D images have never been so easy. The next step of this project is to add body recognition to add more interactions to the application with the Microsoft Kinect. The last version of the project will be sent to a team in Portugal, which will continue to work and improve the concept to make it usable in a medical context.

Leap motion, Kinect, 3D interaction, medical data's, python

## Résumé :

Les systèmes actuels de visualisation de données médicales ne permettent pas une manipulation simple pour les médecins. Le but de ce projet était de simplifier les interactions et de les rendre sans contact. L'idée de départ était de développer une nouvelle méthode d'interaction avec les images électroniques (IRM, rayons X, etc ). D'autres projets sur ce sujet ont déjà été réalisés, mais pas avec le matériel disponible pour ce projet. Un traqueur de main infrarouge nommé Leap Motion a été utilisé pour récupérer la position, l'orientation et le nombre de mains et de doigts. Avec ces données, les différents codes sources ont été écrits en Python pour analyser les mouvements et «enseigner» les interactions humaines à l'ordinateur. Après cela, la dernière chose à programmer était de relier les mouvements des mains aux mouvements des objets virtuels sur l'ordinateur. Les images que nous pouvons manipuler représentent le cerveau, l'estomac ou de tout autre organe que l'on veut observer. Le matériel utilisé ( Leap Motion) est beaucoup plus précis que d'autres matériels utilisés avant. Les résultats sont pour l'instant très positifs et le contrôle de ces images 3D n'a jamais été aussi simple. La prochaine étape de ce projet est d'ajouter la reconnaissance du corps de l'utilisateur pour ajouter plus d'interactions à l'application avec la Kinect de Microsoft. La dernière version du projet sera envoyée à une équipe au Portugal, qui va continuer à travailler et améliorer le concept pour le rendre utilisable dans un contexte médical.

Leap motion, Kinect, interaction 3D, donnée médicale, python