

Filière Systèmes industriels

Orientation Power & Control

Diplôme 2014


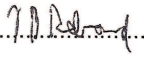
Jean-Baptiste Rebord

*Application d'un "Leap Motion"
à l'apprentissage de trajectoires*

- Professeur
Jean-Daniel Marcuard
- Expert
Bryan Gosparini
- Date de la remise du rapport
11.07.2014

<input checked="" type="checkbox"/> FSI <input type="checkbox"/> FTV	Année académique / Studienjahr 2013/14	No TD / Nr. DA pc/2014/77
Mandant / Auftraggeber <input checked="" type="checkbox"/> HES—SO Valais <input type="checkbox"/> Industrie <input type="checkbox"/> Etablissement partenaire <i>Partnerinstitution</i>	Etudiant / Student Jean-Baptiste Rebord Professeur / Dozent Jean-Daniel Marcuard	Lieu d'exécution / Ausführungsort <input checked="" type="checkbox"/> HES—SO Valais <input type="checkbox"/> Industrie <input type="checkbox"/> Etablissement partenaire <i>Partnerinstitution</i>
Travail confidentiel / vertrauliche Arbeit <input type="checkbox"/> oui / ja ¹ <input checked="" type="checkbox"/> non / nein	Expert / Experte (données complètes) Bryan Gosparini Crevoisier SA 2714 Les Genevez	

Titre / Titel <p style="text-align: center;">Application d'un "Leap Motion" à l'apprentissage de trajectoires</p>
Description / Beschreibung <p>Le "Leap Motion" est un accessoire d'acquisition pour PC capable de reconstituer la trajectoire en 3D des doigts et de la main d'un opérateur. Il est envisageable de l'utiliser pour réaliser rapidement l'apprentissage de mouvements destinés ensuite à être reproduits par un robot ou pour commander un système mobile en temps réel (drone...).</p> <p>Dans cette optique, on peut envisager d'utiliser conjointement plusieurs Leap Motion pour augmenter la précision et/ou la sécurité du système.</p> Objectifs / Ziele Le but du projet de diplôme consiste à : <ul style="list-style-type: none"> — développer et valider une application basée sur un « leap motion » permettant de piloter les mouvements d'un robot en temps différé et – si possible – en temps réel (télémanipulation) — caractériser la précision des mouvements que l'application ci-dessus permet de réaliser — développer dans la mesure du temps disponible une application permettant de piloter un drone.

Signature ou visa / Unterschrift oder Visum Responsable de l'orientation Leiter der Vertiefungsrichtung: 	Délais / Termine Attribution du thème / Ausgabe des Auftrags: 12.05.2014 Remise du rapport / Abgabe des Schlussberichts: 11.07.2014, 12:00 Expositions / Ausstellungen der Diplomarbeiten: 27 – 29.08.2014 Défense orale / Mündliche Verfechtung: Semaine Woche 36
¹ Etudiant / Student : 	

¹ Par sa signature, l'étudiant-e s'engage à respecter strictement la directive DI.1.2.02.07 liée au travail de diplôme.
 Durch seine Unterschrift verpflichtet sich der/die Student/in, sich an die Richtlinie DI.1.2.02.07 der Diplomarbeit zu halten.

Application d'un "Leap Motion" à l'apprentissage de trajectoires

Diplômant/e Jean-Baptiste Rebord



Objectif du projet

Le but de ce travail est de capturer les mouvements de la main au moyen d'un « Leap Motion Controller » et de reproduire ses gestes en direct ou en différé sur un bras robot TX60 de Staubli.

Méthodes | Expériences | Résultats

Le « Leap Motion Controller » est un petit périphérique capable de reconnaître les mouvements des mains placées au-dessus de lui. Il envoie à un ordinateur par câble USB leurs positions et orientations.

Un programme réalisé en Java traite en temps réel les informations reçues par le Leap Motion puis les envoie à la commande du bras robot TX60 qui, après un dernier traitement pour éviter tout risque de collision, reproduit alors le déplacement. Le système est également capable d'enregistrer un mouvement de la main et de le rejouer en différé sur le bras robot.

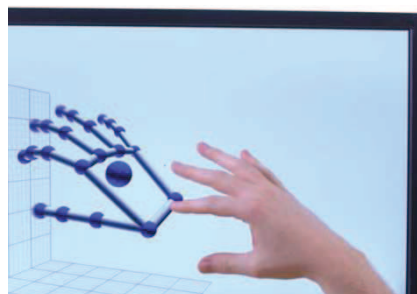
Une pince fixée à l'extrémité du robot permet de saisir des objets. Il est possible de contrôler l'ouverture et la fermeture de cet outil avec le pouce. De plus la pince peut être dirigée vers le bas où vers l'avant.

Travail de diplôme
| édition 2014 |

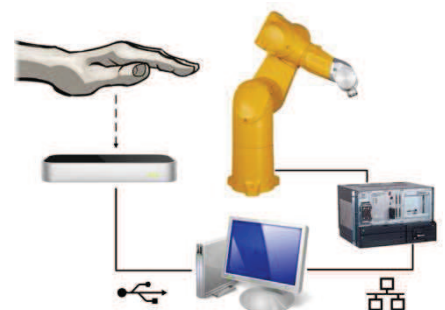
Filière
Systèmes industriels

Domaine d'application
Power & Control

Professeur responsable
Jean-Daniel Marcuard
mad@hevs.ch



Le « Leap Motion Controller » reconnaît les mains placées au-dessus de ses capteurs.



La position et l'orientation de la main sont transmises à un ordinateur qui traite ces données et les renvoie au contrôleur CS8C du bras robot.

Table des matières

1	Introduction.....	7
2	Présentation du Leap Motion Controller	7
2.1	Concurrence et partenariats	10
2.2	Avis sur le Leap Motion	11
2.2.1	Avantages	11
2.2.2	Désavantages.....	11
2.3	Avenir du périphérique	11
3	Mise en service.....	12
3.1	Problèmes de détection version 1.....	12
3.1.1	Instabilité de détection	12
3.1.2	Mauvaise interprétation.....	13
3.1.3	Objets inconnus.....	13
3.1.4	Occlusion partielle.....	13
3.2	Version 2.0 beta	13
4	Environnement de développement avec Java	14
4.1	Nouveau projet.....	14
4.2	Description des méthodes principales	15
4.2.1	onInit	15
4.2.2	onExit.....	15
4.2.3	onConnect	15
4.2.4	onDisconnect.....	15
4.2.5	onFocusGained	15
4.2.6	onFocusLost.....	15
4.2.7	onFrame	15
4.3	Informations reçues	16
4.4	Objets détectés	16
4.4.1	Hand	17
4.4.2	Finger.....	17
4.4.3	Tool.....	17

4.4.4	Pointable.....	17
4.4.5	Gesture	17
4.5	Méthodes des bibliothèques Leap Motion en Java.....	19
4.5.1	Liste des méthodes.....	19
4.5.2	Tests.....	19
4.6	Premier programme: LM Infos	19
4.7	Second programme: LM Capture	20
4.7.1	Mode direct	20
4.7.2	Mode enregistrement	21
4.7.3	Mode lecture	21
4.8	Version 2.0 beta	21
5	Présentation du bras robot Staubli	22
5.1	Caractéristiques du TX60.....	22
5.2	Caractéristiques du contrôleur CS8C.....	23
5.3	Modes de marche.....	24
5.3.1	Mode manuel	24
5.3.2	Mode local	24
5.3.3	Mode déporté	24
5.4	Boîtier de commande manuelle MCP.....	24
5.5	Programmation du TX60	25
5.6	Positions du TX60	25
5.6.1	Positions articulaires	25
5.6.2	Positions cartésiennes	25
5.7	Mouvements du TX60	25
5.7.1	Mouvements manuels.....	25
5.7.2	Mouvements programmés.....	26
6	Réalisation	27
6.1	Principe.....	27
6.2	Fonctionnalités	28
6.2.1	Pince	28
6.2.2	Arrêt et reprise du suivi.....	29
6.2.3	Position relative.....	29
6.3	Java.....	29

6.3.1	Classes	29
6.3.2	Matrices.....	30
6.3.3	Vérifications et limitations	33
6.3.4	Client TCP.....	33
6.3.5	Raccourcis clavier	34
6.4	VAL3.....	35
6.4.1	Matériel	35
6.4.2	Programmes VAL3	35
6.5	Déplacements et fluidité du système.....	37
6.5.1	Repère	37
6.5.2	Configuration.....	37
6.5.3	Fluidité du mouvement	38
6.6	Problèmes rencontrés	39
6.6.1	Collisions.....	39
6.6.2	Retard	39
6.6.3	Variation de la vitesse de lecture	39
6.6.4	Limites des rotations	39
6.6.5	Client TCP.....	40
7	Caractérisation du Leap Motion Controller.....	40
7.1	Première session	41
7.1.1	Méthode	41
7.1.2	Discussion	43
7.2	Deuxième session.....	43
7.2.1	Méthode.....	43
7.2.2	Discussion	49
8	Utilisation de deux Leap Motion Controller.....	50
9	Applications.....	50
9.1	Téلمانipulation avec un bras robot	50
9.2	Leap Motion	51
10	Conclusion	52
11	Références.....	53
12	Annexes	54

1 Introduction

Dans le cadre du projet de semestre réalisé entre février et mai 2014, j'avais la tâche de mettre en service un Leap Motion Controller. L'objectif final était de maîtriser ce périphérique à travers un environnement de développement, de mesurer sa précision et de connaître ses limites afin d'en ressortir des applications intéressantes.

Pour mon travail de diplôme, je dois réaliser au moins une de ces applications. L'une d'elles a pour but de pouvoir faire de la télémanipulation avec un bras robot en utilisant le Leap Motion comme capteur. L'extrémité du bras doit suivre la position de la main de l'utilisateur en tenant compte des trois dimensions. L'orientation doit également être reproduite. Le robot doit pouvoir suivre la main en direct, mais il doit également être en mesure de rejouer en différé les mouvements enregistrés. Les deux programmes du système, en Java sur l'ordinateur et en VAL3 sur la commande du robot, s'assurent du traitement des informations afin d'éviter d'éventuelles collisions.

Le Leap Motion permet également de reconnaître des gestes spécifiques. Ces fonctionnalités ont été mises à profit dans ce projet. Le robot ayant été équipé d'une pince, l'utilisateur a la possibilité par exemple de changer l'état de la pince d'un mouvement du pouce.

Afin de définir dans quelle marge le Leap Motion Controller pourrait être utilisé dans l'industrie, la caractérisation du Leap Motion Controller a été réalisée. Pour cela, des mesures de position sont effectuées, puis comparées à une référence. Les erreurs sont alors calculées, et les résultats sont analysés.

2 Présentation du Leap Motion Controller

Le Leap Motion Controller a été développé par l'entreprise Leap Motion Inc. fondée en 2010 par Michael Buckwald et David Holz en Californie aux USA. C'est actuellement le seul produit vendu par l'entreprise.

Le Leap Motion Controller est un périphérique USB qui joue le rôle d'interface entre un ordinateur et un utilisateur au même titre qu'un clavier ou une souris.



Figure 1 : Leap Motion Controller

Physiquement, il se présente sous la forme d'un boîtier aux dimensions modestes (80 x 30 x 11 mm). Sur l'un des côtés se trouve une sortie USB (compatible avec USB2 et USB3) pour le connecter à un ordinateur (compatible pour Windows, Mac et Linux). Une vitre noire couvre sur le dessus de l'appareil. On peut y distinguer trois LEDs infrarouges à l'intérieur. Derrière la vitre se trouvent

également deux capteurs optiques de 1.3 mégapixels qui détectent ce que les LEDs éclairent en prenant en compte les trois dimensions.

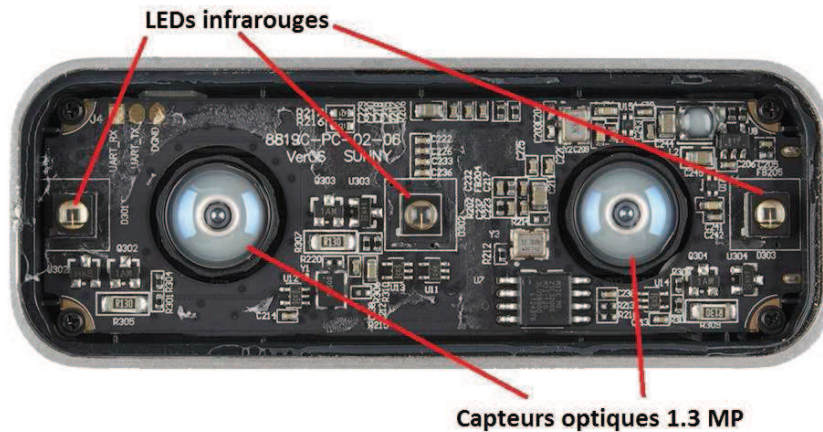


Figure 2 : Intérieur du Leap Motion Controller

Pour l'utiliser, il faut le poser à plat sur la table devant son clavier. Son interaction avec l'utilisateur se fait par la détection de la position et des mouvements des mains placées au-dessus de lui dans une demi-sphère d'un rayon de 61 cm. La position et la direction de chaque doigt sont également détectées.

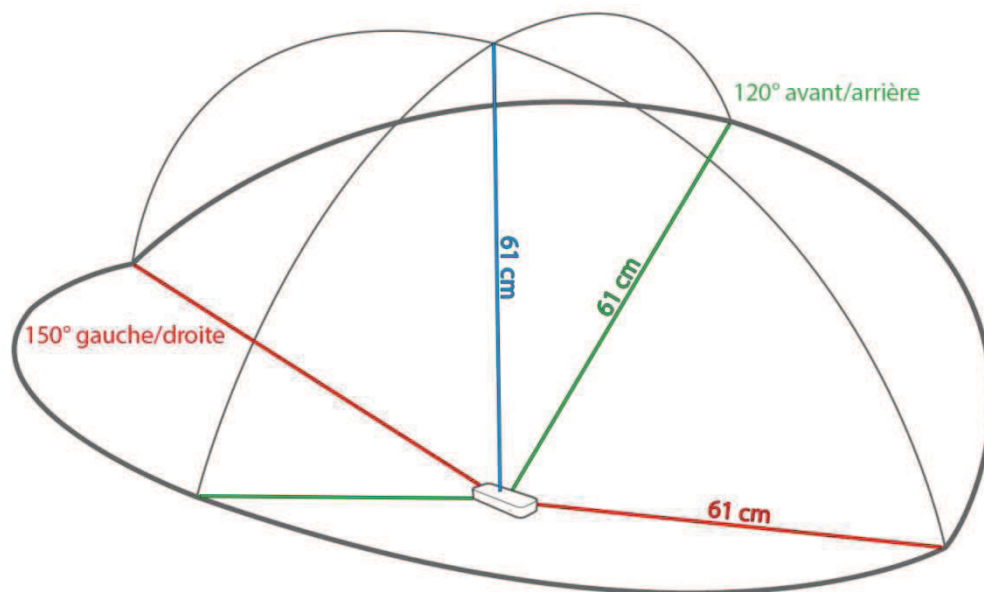


Figure 3 : Zone de détection du Leap Motion Controller

Pour fonctionner, le pilote officiel doit être installé sur l'ordinateur. Les données sur les objets perçus par le capteur sont immédiatement envoyées par le câble USB et le pilote interprète ces informations pour les programmes.

Ce périphérique est sorti le 22 juillet 2013. Son prix de vente en Europe est de 89.99 €. En Suisse, il est vendu 99.95 CHF. Les développeurs qui le souhaitaient avaient la possibilité d'obtenir une version du produit avant sa mise officielle sur le marché. Le Leap Motion Controller a connu plusieurs évolutions lors de son développement.

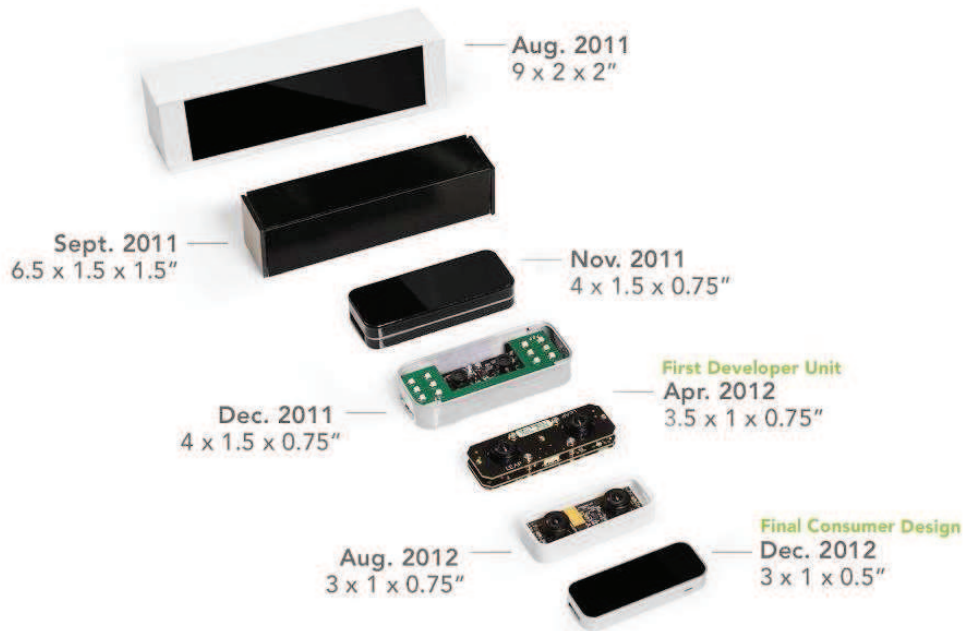


Figure 4 : Evolution du développement du Leap Motion Controller

Beaucoup de développeurs particuliers ont vu un grand potentiel dans ce nouveau périphérique. Très vite, des programmes utilisant cette interface ont fait leur apparition sur la page des développeurs du site officiel du contrôleur. Lesdits programmes sont accompagnés de leur code source complet. Beaucoup de langages de programmation sont représentés (C, C++, Java, JavaScript, Python, Ruby...).

Le site officiel du Leap Motion Controller propose également un espace de téléchargement de programmes appelé Airspace Store. Celui-ci a fait son apparition quelques mois avant la commercialisation du produit. Il vise toutes les personnes acquérant leur produit et propose des applications gratuites aussi bien que payantes. Leur prix peut varier entre quelques cents à plusieurs centaines de dollars. La qualité des programmes présents sur cette plate-forme est contrôlée par l'entreprise. De nombreuses catégories sont représentées (jeux, éducation, création, musique, science...).

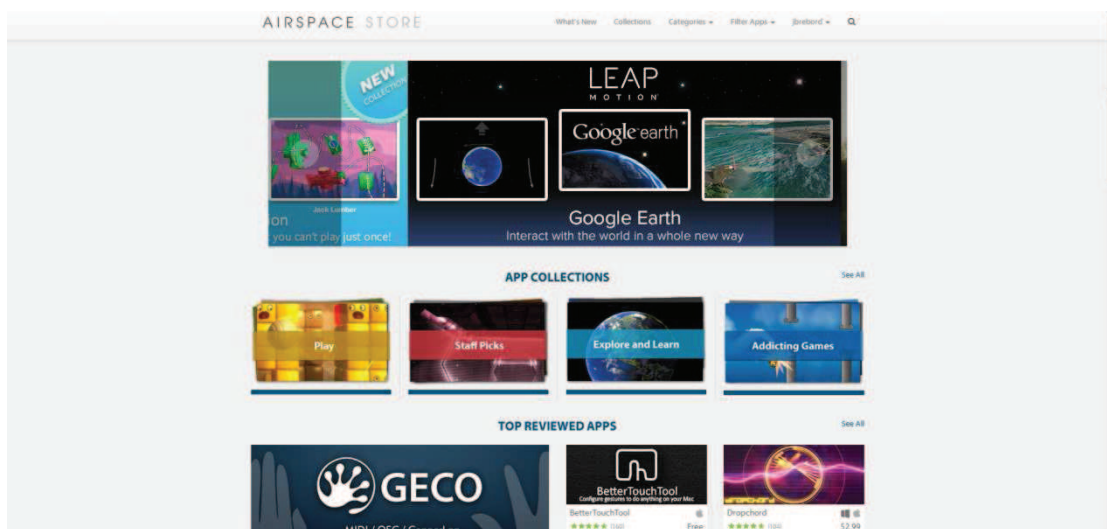


Figure 5 : Plate-forme de téléchargement

Des programmes existants déjà avant la sortie du Leap Motion Controller se sont vus dotés de fonctionnalités leur permettant d'être pilotés par ce périphérique. Google Earth, par exemple, est maintenant pilotable sans clavier ni souris.

Il est impossible d'utiliser exclusivement le Leap Motion Controller sur un ordinateur. Il ne peut pas remplacer complètement le clavier, mais il apporte une dimension supplémentaire. Des applications existantes sur Airspace Store permettent de contrôler le curseur de la souris avec le Leap Motion. Les actions sont cependant compliquées, parfois instables et surtout, plus lentes à réaliser.

2.1 Concurrence et partenariats

Vu le potentiel ludique de cette interface, il est normal de vouloir la comparer au capteur Kinect. Ce dernier est un périphérique fabriqué par Microsoft qui se branche sur leur console de salon, la Xbox, ou sur un PC. Il permet de détecter la position et les mouvements des personnes se trouvant devant lui. Sorti en 2010, il a connu un succès commercial très important. Même si le principe reste très similaire au Leap Motion Controller, à savoir capter des positions en trois dimensions, l'utilisation n'est pas la même. Le capteur Kinect a son objectif sur l'un des côtés et cherche des cibles plutôt grandes (de la taille d'une personne) et assez éloignées de lui alors que le Leap Motion Controller se pose à plat, il détecte des mains et sa zone de détection est beaucoup plus restreinte. Une version 2 du capteur Kinect est sortie en fin 2013.



Figure 6 : Kinect

Plusieurs partenariats ont déjà été annoncés. Asus a indiqué que le contrôleur serait intégré dans leurs ordinateurs portables haut de gamme et leurs PC tout-en-un.

L'entreprise HP, également en partenariat avec Leap Motion Inc., vend aujourd'hui plus de 10 modèles d'ordinateurs portables avec le capteur intégré devant le clavier.



Figure 7 : Ordinateur HP équipé d'un Leap Motion Controller

2.2 Avis sur le Leap Motion

Le Leap Motion Controller divise clairement les développeurs. Certains voient dans cet objet un grand nombre d'opportunités de développement. Mais il présente aussi de nombreux points faibles qu'il ne faut pas sous-estimer.

2.2.1 Avantages

Temps de réaction très faible
Volume du boîtier modeste
Prix très bas
Bonne compatibilité avec les différents OS
Certains programmes gratuits

2.2.2 Désavantages

Problèmes de détection
Prise en main non intuitive
Pas forcément meilleur que ce qui existe déjà (clavier, souris)
Fatigue des bras

2.3 Avenir du périphérique

Les ventes du Leap Motion Controller ont été très décevantes pour l'entreprise. Alors que celle-ci s'attendait à vendre plus de 5 millions d'unités, ce n'est que 500'000 périphériques qui ont trouvé preneur. Cela a conduit au licenciement de 10% des 120 employés de l'entreprise afin de permettre l'engagement d'ingénieurs supplémentaires dans l'optique d'améliorer le produit.

A la fin mai 2014, soit moins d'un an après la mise en vente officielle de l'interface, un nouveau pilote en version beta fait son apparition sur la toile à disposition des développeurs. Au programme, une détection plus stable et plus précise : chaque main et chaque doigt peuvent être étiquetés (main droite/gauche, pouce index...), les phalanges sont différenciées, les éventuelles sources infrarouges externes sont mieux compensées. La société est repartie d'un bon pied pour corriger la fragilité de la première version.

3 Mise en service

Après avoir installé le pilote du périphérique et branché celui-ci au PC, l'utilisateur peut immédiatement faire fonctionner le Leap Motion Controller. Un outil de visualisation est proposé dans le logiciel accompagnant le pilote. Le programme Airspace, plate-forme de téléchargement, est également installé automatiquement.

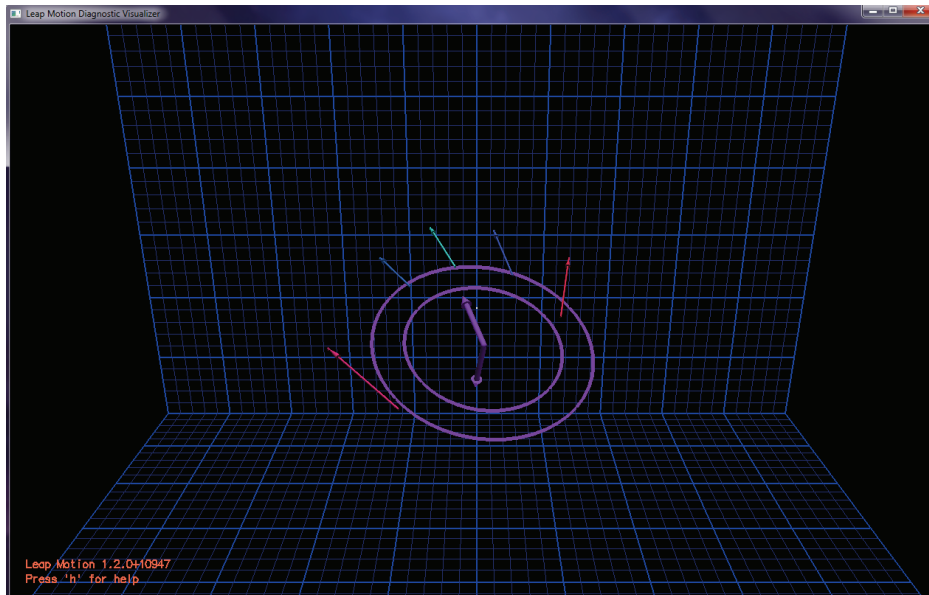


Figure 8 : Visualiseur de diagnostics avec détection d'une main (Version 1.2)

En essayant l'outil de visualisation et en plaçant sa main au-dessus du capteur, on remarque que celle-ci est représentée comme un cercle avec deux vecteurs partant de son centre, un vers le bas et l'autre vers l'avant. Les doigts, quant à eux, sont de simples vecteurs.

L'utilisation du Leap Motion Controller avec Google Earth nécessite un temps d'adaptation avant qu'il ne soit possible de bien comprendre à quel geste correspond quel déplacement. Chaque programme possède ses propres règles de guidage et celui-ci n'est pas forcément intuitif. Bien que l'on retrouve certaines mécaniques communes, il n'existe pas de normes dans ce domaine. De plus, l'expérience sera complètement différente selon le type d'affichage (par exemple selon le nombre de dimensions). C'est une difficulté supplémentaire pour le développeur de faire en sorte que l'utilisateur s'y retrouve le plus facilement possible.

3.1 Problèmes de détection version 1

On constate de nombreuses interférences à l'aide du visualiseur de diagnostics fourni avec le pilote du produit:

3.1.1 Instabilité de détection

Souvent, sans aucune raison, le capteur ne détecte plus ou mal l'un des objets présentés au-dessus de lui. Une luminosité infrarouge externe trop intense ou le rapprochement de deux objets (par exemple 2 doigts) peut être la cause de ce problème.

3.1.2 Mauvaise interprétation

Le périphérique a également du mal à détecter dans quel sens est positionnée une main si les doigts sont tous tendus. En effet, si l'on effectue une rotation de 180° sur l'axe de roulis de la main, le Leap Motion Controller interprétera souvent ce mouvement comme une rotation de 90° suivie d'une autre rotation de 90° dans le sens opposé. Placer deux périphériques peut résoudre ce problème.

3.1.3 Objets inconnus

Si quelqu'un place une main devant lui au-dessus du contrôleur, et que sa tête est un peu trop en avant, le périphérique risque d'interpréter ce nouvel élément comme une main. On constate avec le visualiseur que la position de cet "objet indésirable" n'est pas stable (la main affichée vibre). La solution à ce problème consiste à programmer un filtre en limitant par exemple le nombre d'objets à détecter ou en rapetissant le volume de détection.

3.1.4 Occlusion partielle

Un objet situé derrière un autre est invisible pour le capteur. Typiquement, si une main est placée de profil doigts tendus, la position des rares doigts détectés ne sera pas stable. Une solution pour remédier à ce problème est de placer un deuxième Leap Motion Controller sous un angle différent. Ainsi, si un objet est caché pour un capteur, il sera visible pour le second.

3.2 Version 2.0 beta

Le visualiseur de diagnostics a évolué dans la version beta du nouveau pilote. Les articulations sont maintenant prises en compte. Les avant-bras sont également reconnus.

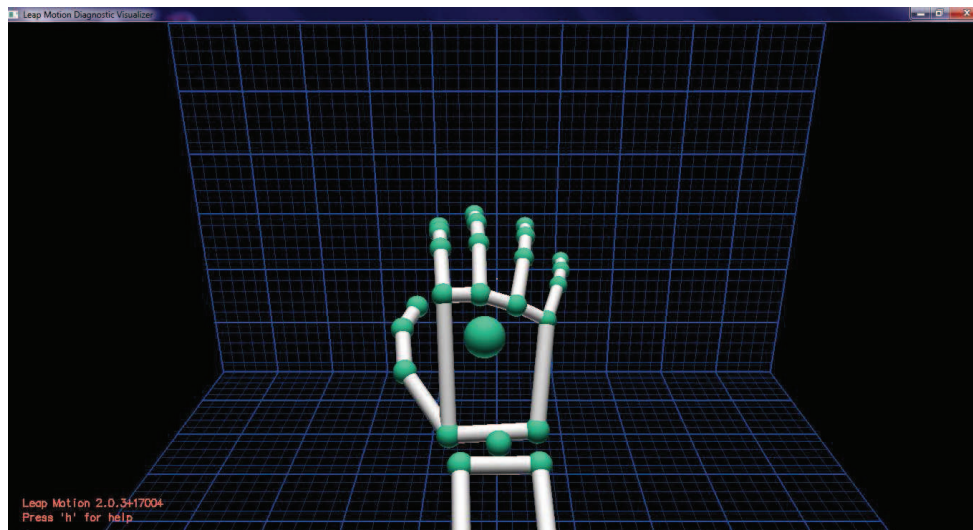


Figure 9: Visualiseur de diagnostics avec détection d'une main (Version 2.0)

Les problèmes cités précédemment sont en grande partie corrigés avec cette mise à jour:

-La détection est plus stable et les objets ont moins tendance à disparaître. En effet, la main est dorénavant toujours affichée avec cinq doigts même si ceux-ci ne sont pas détectés. Ce qui n'était pas le cas avec la première version du pilote où ils disparaissaient.

-La rotation d'une main de 180° est maintenant toujours interprétée correctement. Comme le Leap Motion cherche à savoir s'il s'agit d'une main gauche ou d'une main droite, il n'est plus possible de le tromper en retournant celle-ci. Cependant, si l'on entre dans son champ de vision la paume vers le

haut avec les doigts dépliés, il interprétera souvent cela comme l'autre main la paume dirigée vers le bas.

-La tête de l'utilisateur n'est plus interprétée comme une main instable.

-Le problème de l'occlusion partielle est également résolu, car une main s'affichera toujours avec cinq doigts. Si l'un d'eux venait à être caché, le Leap Motion calculera sa position la plus probable. De ce fait, si la main est placée de profil par rapport au Leap Motion, aucun doigt ne sera affiché.

4 Environnement de développement avec Java

Durant mon projet de semestre, l'un des principaux objectifs était de me familiariser avec l'environnement de développement. J'ai choisi d'utiliser le langage de programmation Java qui est un langage orienté objets. La version du JDK utilisée est la 1.7.55. Les programmes ont été écrits sur le logiciel eclipse version 4.2.1.

Afin de créer un programme qui prend en compte les données envoyées par le Leap Motion Controller, il faut s'inscrire dans la section des développeurs sur le site officiel et télécharger le pack contenant les bibliothèques des différents langages de programmation. En créant un nouveau projet Java, il faut importer dans celui-ci les bibliothèques relatives au Leap Motion.

4.1 Nouveau projet

Si l'on veut pouvoir prendre en compte les données envoyées par le Leap Motion Controller dans un programme en langage java, il faut d'abord instancier le contrôleur ainsi qu'un objet "listener" dont la classe "MyListener" contient les méthodes des événements envoyés par le périphérique. Les deux nouveaux objets doivent ensuite être liés:

```
1 Controller controller = new Controller();
2 MyListener listener = new MyListener();
3 controller.addListener(listener);
```

La classe MyListener créée doit hériter de la classe mère Listener contenue dans les bibliothèques Leap Motion:

```
class MyListener extends Listener
```

Cette classe peut contenir les méthodes principales suivantes:

```
1 public void onInit(Controller controller){
2 }
3 public void onExit(Controller controller){
4 }
5 public void onConnect(Controller controller){
6 }
```

```
7 public void onDisconnect (Controller controller) {  
8 }  
9 public void onFocusGained (Controller controller) {  
10 }  
11 public void onFocusLost (Controller controller) {  
12 }  
13 public void onFrame (Controller controller) {  
14 }
```

4.2 Description des méthodes principales

4.2.1 onInit

Méthode exécutée au moment où l'objet listener est attaché au controller.

4.2.2 onExit

Méthode opposée de onExit, elle est exécutée lorsque l'objet listener est détaché du controller.

4.2.3 onConnect

Méthode exécutée au moment où le programme se connecte au périphérique. Elle est également appelée lorsque le Leap Motion est branché à l'ordinateur après l'exécution du programme. C'est dans cette méthode que doivent être déclarés les gestes que l'on souhaite prendre en compte.

4.2.4 onDisconnect

Méthode opposée de onConnect, elle est exécutée lorsque le programme se déconnecte du périphérique ou lorsque ce dernier est débranché de l'ordinateur.

4.2.5 onFocusGained

Méthode appelée lorsque le programme utilisant le Leap Motion obtient le focus.

4.2.6 onFocusLost

Méthode opposée de onFocusGained, elle est exécutée lorsque le programme perd le focus.

4.2.7 onFrame

C'est la méthode la plus importante. Il est en effet possible de faire fonctionner le Leap Motion dans un programme en langage java en utilisant uniquement cette méthode de la classe MyListener. Elle est exécutée chaque fois qu'une nouvelle frame est envoyée par le contrôleur à l'ordinateur. La fréquence d'échantillonnage peut aller jusqu'à plus de 200 FPS (frames par seconde) sur un port USB3. Avec un ordinateur disposant d'un port USB2, la fréquence la plus courante obtenue est d'environ 115 FPS. Lorsqu'aucun objet n'est présent au-dessus du périphérique, la fréquence d'échantillonnage tombe à 30 FPS.

Une frame est une "image" qui contient toutes les informations concernant les différents objets détectés par le capteur.

4.3 Informations reçues

Chaque objet détectable possède un identifiant (id). Celui-ci permet de différencier chaque objet. Il est attribué automatiquement dès que l'objet est détecté et reste le même aussi longtemps que l'objet est visible par le capteur.

Les valeurs des positions sont données en millimètres sous la forme de trois valeurs représentant chacune une des trois dimensions.

Les vecteurs sont également sous forme de trois composantes, une pour chaque dimension. La norme de chaque vecteur est toujours égale à 1.

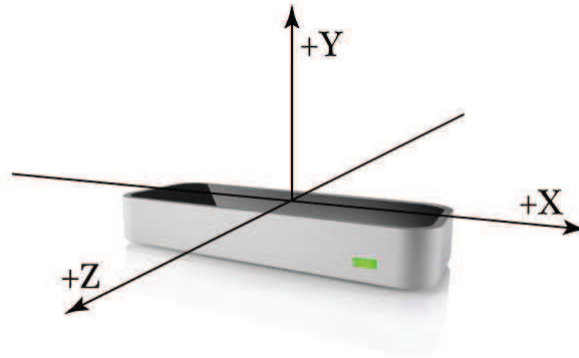


Figure 10 : Axes 3D du Leap Motion Controller

4.4 Objets détectés

Pour obtenir à chaque nouvelle image les informations sur les objets détectables, il faut créer, dans la méthode onFrame, un objet Frame qui appelle la méthode frame() de la classe Controller.

```
Frame frame = controller.frame();
```

Cet objet frame contient les listes de tous les objets visibles par le Leap Motion Controller ainsi que toutes leurs informations respectives.

Chaque objet détectable possède sa méthode correspondante qui retourne la liste de tous les objets de même nature. Par exemple, la méthode hands() de la classe HandList permet d'obtenir des informations sur toutes les mains détectées.

L'indexation des objets dans leur liste respective est automatique: s'il y a deux mains visibles et que la première est retirée du champ de vision du capteur, l'autre prend sa place. L'id de cette dernière va cependant rester la même.

Pour spécifier que l'on souhaite des informations relatives à une main clairement déterminée, on utilise la méthode get() en entrant comme paramètre le numéro de cette main.

Par exemple, si l'on désire connaître la position de la première main de la liste, il faut entrer le code suivant:

```
frame.hands().get(0).palmPosition();
```


A noter qu'il est également possible de créer un objet avant de récupérer ses informations. Ainsi on peut obtenir la position de la main 0 avec le code suivant:

```
Hand hand = frame.hands().get(0)
hand.palmPosition();
```

Ces deux extraits retournent un vecteur de cette forme: "(-13.816, 204.315, 21.9744)". A noter que l'objet Vector des bibliothèques du Leap Motion Controller n'est pas identique à celui présent dans les bibliothèques de base de Java. Le premier représente un vecteur en trois dimensions alors que le second est un tableau d'objets.

Les coordonnées du vecteur peuvent être récupérées individuellement en spécifiant l'axe en fin de ligne:

```
frame.hands().get(0).palmPosition().getX();
```

Ce code renvoie un nombre de type float.

Les règles données précédemment pour obtenir des informations sur les mains sont similaires pour les autres objets détectables. Les objets pouvant être captés par le Leap Motion Controller sont les suivants:

4.4.1 Hand

Une main est représentée par une position et deux vecteurs perpendiculaires: l'un d'eux suit une direction normale au plan que forme la paume et l'autre suit la direction de la main.

4.4.2 Finger

Un doigt est représenté par un point et une direction. Ce qui forme simplement un vecteur allant dans la direction que pointe le doigt. Le contrôleur donne aussi des informations sur sa longueur et sa largeur.

4.4.3 Tool

Un outil retourne exactement les mêmes informations qu'un doigt. Il s'agit aussi d'un simple vecteur. Le Leap Motion Controller peut différencier les deux objets d'après la longueur et la largeur de ceux-ci, mais il lui arrive de les confondre.

4.4.4 Pointable

Un pointable regroupe les doigts et les pointables sous une même appellation si l'application du programme développé l'exige.

4.4.5 Gesture

Un geste est un objet particulier, car il n'a pas d'équivalent matériel. C'est le mouvement de la main et des doigts qui va créer un objet geste. Cet objet est présent un court instant lors de sa détection avant de disparaître. Il y a quatre gestes qui sont déjà enregistrés dans les bibliothèques du Leap Motion

et qu'il suffit d'activer au début du programme (méthode onConnect) pour être détectables. Cette opération ne doit pas être négligée, car le geste n'apparaîtra pas dans les objets s'il n'a pas été déclaré lorsque le programme démarre.

Les quatre gestes existants dans la version 1.0 des bibliothèques sont:

4.4.5.1 Circle

Ce geste est détecté lorsque l'on trace un cercle dans l'air. Le Leap Motion Controller peut renvoyer le rayon du cercle tracé ainsi que des informations sur le centre de ce cercle.

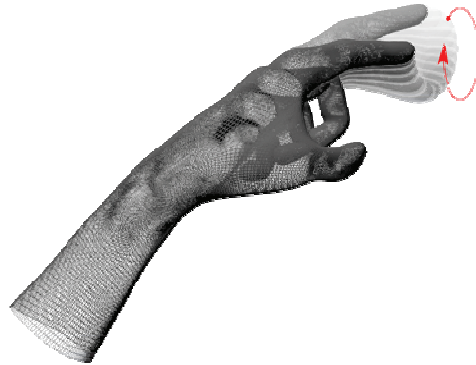


Figure 11 : Geste Circle

4.4.5.2 Swipe

Un geste de va et vient d'une main placée de profil par rapport au capteur.



Figure 12 : Geste Swipe

4.4.5.3 Key Tap

Un geste qui représente la pression sur un bouton, mais celui-ci est effectué dans l'air au-dessus du contrôleur.



Figure 13 : Geste Key Tap

4.4.5.4 Screen Tap

Un geste où un pointable traverse le plan vertical parallèle à l'écran passant par le centre du périphérique. Le programme est capable de récupérer les coordonnées de l'intersection du pointable et du plan.

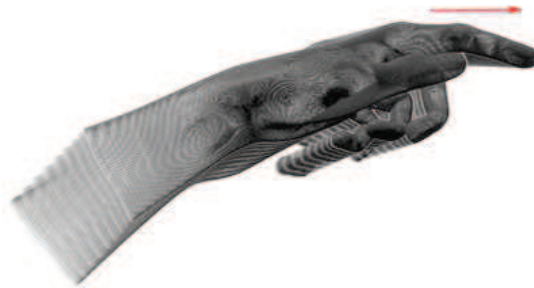


Figure 14 : Geste Screen Tap

Si l'on souhaite compléter cette liste, il est possible de créer soi-même un nouveau geste en écrivant le code qui lui correspond.

4.5 Méthodes des bibliothèques Leap Motion en Java

4.5.1 Liste des méthodes

La liste des méthodes testées et utilisées pour mon projet se trouve en annexe (Annexe 1).

4.5.2 Tests

Toutes les lignes de code décrites en annexe ont été testées et fonctionnent parfaitement. Le temps de réaction est extrêmement faible: à peine une main entre-t-elle dans le champ du contrôleur que toutes ses caractéristiques apparaissent dans le programme.

4.6 Premier programme: LM Infos

Réalisé dans le cadre du projet de semestre, ce premier programme a comme but de connaître l'étendue des informations fournies par le Leap Motion Controller.

Visuellement, le programme se présente sous la forme d'une fenêtre où sont affichées par colonnes des informations sur les objets détectés par le Leap Motion. Celles-ci sont mises à jour à chaque frame.

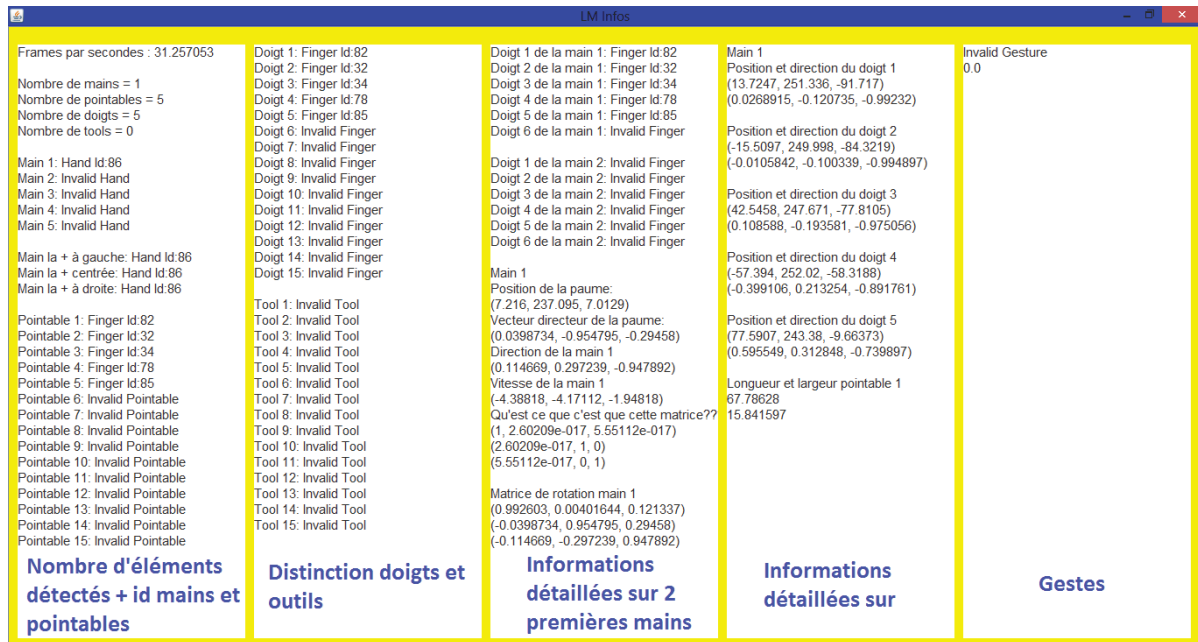


Figure 15 : Premier programme: LM Infos

4.7 Second programme: LM Capture

Toujours dans le cadre du projet de semestre, ce second programme a été réalisé dans le but d'enregistrer les informations des frames afin de caractériser le Leap Motion Controller.

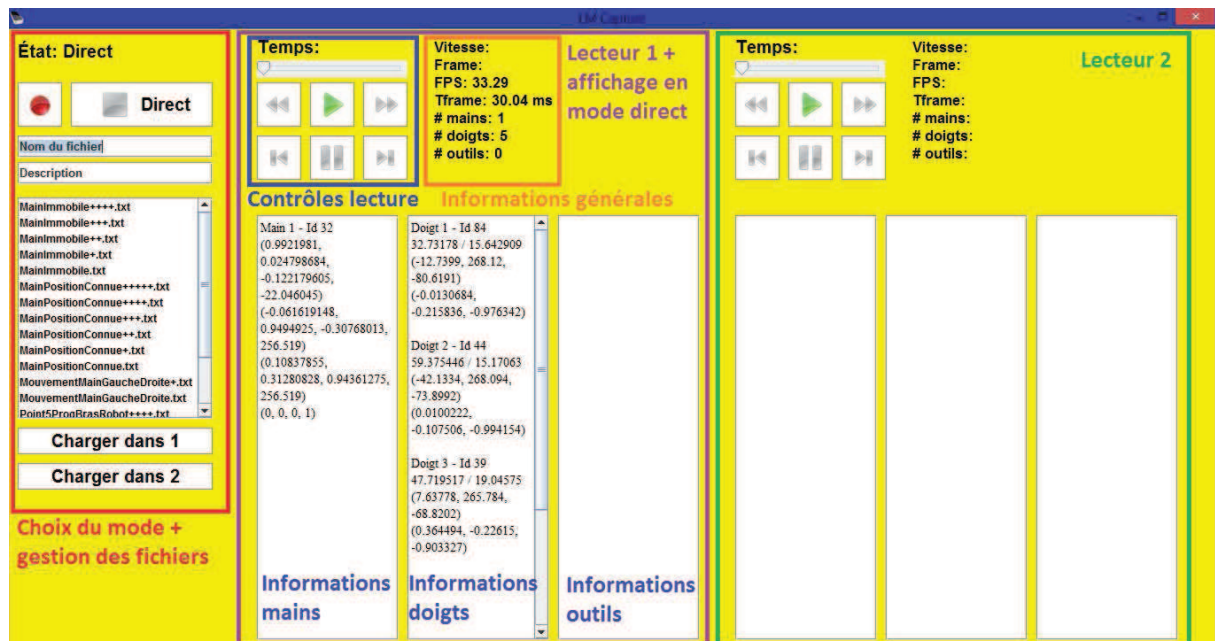


Figure 16 : Second programme: LM Capture

Le programme possède 3 modes:

4.7.1 Mode direct

Les informations sont affichées en direct dans la fenêtre.

4.7.2 Mode enregistrement

Les informations ne sont plus affichées dans la fenêtre. Toutes les frames sont enregistrées dans un fichier txt.

4.7.3 Mode lecture

Le programme permet d'importer un fichier txt créé lors d'un enregistrement et d'afficher le contenu de chaque frame. De plus, le programme possède deux lecteurs dans le but de pouvoir opérer des comparaisons entre différents fichiers.

4.8 Version 2.0 beta

La version beta du nouveau pilote apporte avec elle une mise à jour du SDK avec son lot de nouveautés dans les librairies du Leap Motion. Aucune ancienne méthode n'a été supprimée, ce qui implique une entière compatibilité avec les programmes écrits avant la mise à jour.

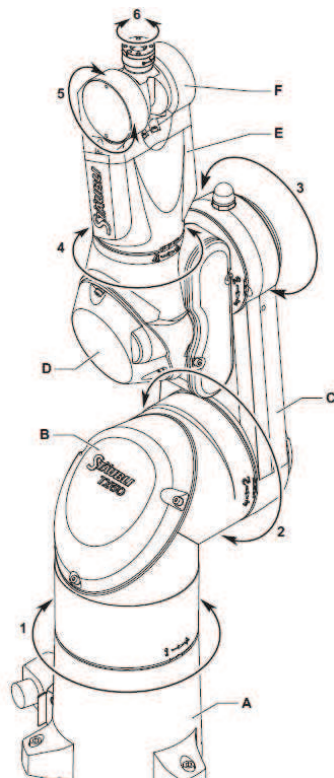
Dans la première version, l'ordre des doigts de FingerList était donné en fonction de l'ordre de détection. Maintenant qu'une main est toujours détectée avec cinq doigts, ceux-ci ont toujours le même ordre. Le doigt 0 correspond donc toujours au pouce, le 1 à l'index et ainsi de suite. De plus, il est maintenant possible de différencier une main gauche d'une main droite.

De nouveaux objets sont détectables: il est possible d'obtenir des informations sur les articulations des doigts ainsi que sur l'avant bras. Les classes des objets déjà existantes dans la version 1.0 se sont vues augmentées de nouvelles méthodes.

5 Présentation du bras robot Stäubli

Le groupe Stäubli International AG, dont le siège social est situé à Pfäffikon en Suisse, a été fondé en 1892. Il est spécialisé dans les domaines textile, connectique et robotique. La société est présente dans 25 pays et occupe plus de 4000 employés.

Le TX60 HB WS est un bras robot 6 axes fabriqué dans l'un des centres industriels du groupe en France. Destiné aux applications machines-outils, sa structure est entièrement capotée afin de pouvoir être employé dans des environnements difficiles. Il peut être fixé au sol, sur un mur ou au plafond. Dans le cadre du projet, le robot est monté sur une armoire. Des codeurs innovants permettent au système de connaître en tout temps la position absolue du bras.



Éléments du bras TX60:

- A: Pied
- B: Epaule
- C: Bras
- D: Coude
- E: Avant-bras
- F: Poignet

Figure 17: Éléments et axes du bras robot TX60

Le robot n'est à la base pas pourvu de pince à son extrémité, mais une interface mécanique en permet la fixation. Un circuit pneumatique et un circuit électrique peuvent assurer la commutation de la pince en cas de besoin.

5.1 Caractéristiques du TX60

- Masse du robot: 51.4 Kg
- Charge nominale: 3.5 Kg
- Charge maximale: 9 Kg
- Rayon d'action: 670 mm
- Répétabilité: ± 0.02 mm
- Classe de protection: IP65
- Classe de protection du poignet: IP67
- Mode de fixation: Sol, mur ou plafond
- Températures de fonctionnement: 5 à 40°C

-Humidité:	30% à 95%
-Altitude:	2000m maximum
-Pression (circuit pneumatique):	1.5 à 7 bar

Ce robot, à l'instar de plusieurs modèles de bras du groupe Stäubli, est mis en mouvement par l'intermédiaire du contrôleur CS8C lequel gère le positionnement des axes et les différentes entrées-sorties extérieures. Le bouton d'alimentation principale pour enclencher le bras robot se trouve sur ce contrôleur. Tout l'équipement est alimenté par du 230VAC et la puissance maximale s'élève à 1.7kVA.



Figure 18: Contrôleur CS8C

5.2 Caractéristiques du contrôleur CS8C

-Dimensions: H x L x D:	331 x 445 x 455 mm
-Classe de protection:	IP20 (IP54 en option)
-Capacité mémoire:	64 MB de RAM (minimum)
-Stockage mémoire:	128 Mo (minimum) Flash Disk et port USB
-Langage de programmation	VAL 3
-Communication:	Liaison série RS232/422 – Serveur Modbus Ethernet
-Entrées/Sorties (E/S):	1 ou 2 cartes 16E/16S digitales (en option)
-Bus de terrain:	DeviceNet, Profibus, CANopen, ModBus, ProfiNet
-Poids:	50 kg

L'utilisateur se sert du boîtier de commande manuelle MCP (Manual Control Pendant) branché sur le CS8C pour contrôler manuellement, paramétrer et exécuter des programmes sur le robot. L'interface opérateur WMS (Working Mode Selection) est également reliée au contrôleur pour sélectionner le mode de fonctionnement du robot. Sur ces deux commandes se trouve un bouton d'arrêt d'urgence.



Figure 19: boîtier de commande manuelle MCP et interface opérateur WMS

5.3 Modes de marche

Il existe trois modes de marche sélectionnables depuis l'interface opérateur :

5.3.1 Mode manuel

Symbole du mode: 

L'utilisateur doit maintenir le bouton de validation ainsi que la touche "move" du boîtier de commande manuelle pour que le robot se déplace.

5.3.2 Mode local

Symbole du mode: 

Une fois le programme lancé et la touche move pressée, le robot exécute des mouvements de façon autonome.

5.3.3 Mode déporté

Symbole du mode: 

Permet à un système externe de mettre le robot sous puissance.

5.4 Boîtier de commande manuelle MCP

Le MCP permet la mise sous puissance et le contrôle des mouvements du bras.

Sur ce boîtier, de nombreux paramètres peuvent être modifiés comme les activations des entrées et sorties, les vitesses maximales du mode manuel et les limites logicielles des articulations. Certains réglages ne peuvent se faire qu'avec un mot de passe.

Si l'on souhaite exécuter une application téléchargée depuis un PC, c'est par cette commande que l'utilisateur doit passer. Il peut également faire bouger manuellement le bras avec des mouvements articulaires ou cartésiens.

Le MCP donne aussi des informations sur l'état de l'installation comme la position articulaire et cartésienne en temps réel, l'état des entrées et sorties, l'état des codeurs et les informations sur la mémoire.

En cas d'erreur, c'est sur l'affichage du boîtier que celle-ci va se manifester. Il faut alors quittancer l'erreur avant de pouvoir entreprendre une nouvelle action. Si l'arrêt d'urgence a été activé, l'utilisateur doit replacer le MCP sur son support afin de permettre à nouveau la mise sous tension du bras.

5.5 Programmation du TX60

La programmation du bras robot s'écrit en VAL3. Langage de haut niveau dédié à la robotique, il permet de contrôler les actions du robot tout en reprenant les caractéristiques des langages de programmation structurés courants.

L'écriture en VAL3 se fait sur le logiciel Stäubli Robotics Suite (SRS) version 2013.0.2. L'utilisateur doit d'abord créer une nouvelle cellule en spécifiant le modèle du robot et du contrôleur CS8C. Ensuite, une nouvelle application doit être créée dans cette cellule. Cette application contiendra tous les programmes. Attention au sens du mot "programme" qui en VAL3 est équivalente à une fonction pour les langages de programmation de haut niveau courants (C, Java..). Un programme est appelé à l'aide de la commande "call" en spécifiant dans les parenthèses les différents paramètres voulus.

Les entrées et sorties que l'utilisateur souhaite prendre en compte dans son programme doivent être activées dans le menu "IO Physiques" de l'onglet "Accueil" de SRS et être liées à une variable VAL3.

5.6 Positions du TX60

Il y a deux moyens de définir un point dans l'espace avec ce bras.

5.6.1 Positions articulaires

Les six données transmises au robot correspondent aux valeurs de chacun des axes.

5.6.2 Positions cartésiennes

Le type trsf de Stäubli est composé de six coordonnées. Trois pour la translation (x, y, z) et trois pour l'orientation (rx, ry, rz).

Ici le point ciblé doit être rattaché à un repère pour que le robot puisse s'y rendre. Le repère par défaut est "world". Son emplacement se trouve dans l'épaule du robot à l'intersection des axes 1 et 2 (Figure 20).

Les trois coordonnées rx, ry et rz, appelées angles d'Euler, correspondent aux angles des rotations qu'il faut appliquer successivement autour des axes X, Y et Z pour obtenir l'orientation du repère.

5.7 Mouvements du TX60

5.7.1 Mouvements manuels

Il existe quatre modes de déplacement manuels du bras robot: joint, frame, tool et point.

Pour un mouvement manuel, le MCP est doté d'un pavé de touches de déplacement qui se présente sous la forme de six paires de boutons -/+. Ce pavé est valable pour les trois premiers modes.

Il est très important de noter que même s'il s'arrête en cas de dépassement des limites matérielles et logicielles des articulations, le bras robot Stäubli est incapable de prévenir une collision et de s'arrêter afin d'empêcher celle-ci.

5.7.1.1 *Mouvements en mode Joint*

On se déplace ici en mode articulaire. Chaque paire -/+ du pavé de touches de déplacement commande une articulation.

5.7.1.2 *Mouvements en mode Frame*

Mode cartésien dans le référentiel de la base du robot (repère world). Trois paires -/+ de touches gèrent la translation de l'extrémité du bras et les trois autres pilotent sa rotation.

5.7.1.3 *Mouvements en mode Tool*

Même principe que le mode précédent sauf que l'utilisateur travaille ici avec l'outil comme référentiel (repère "flange" si aucun outil n'est placé).

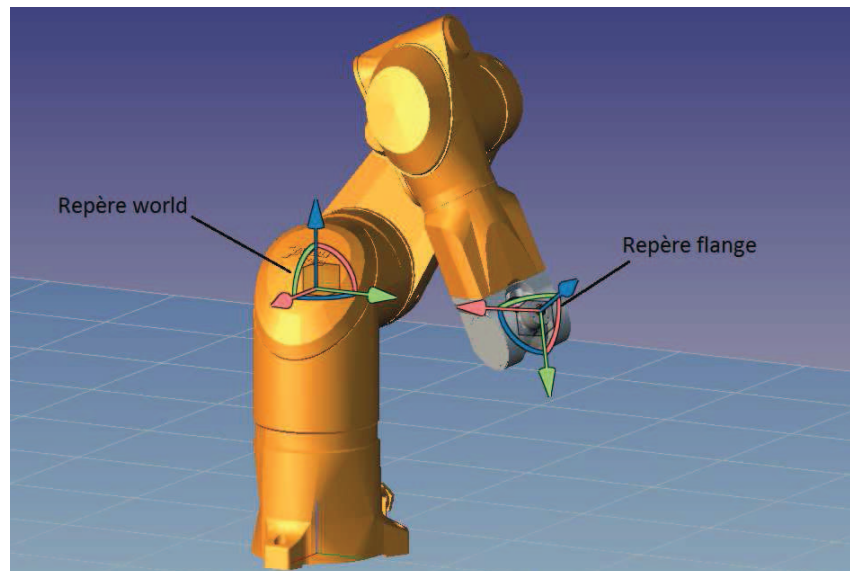


Figure 20: Repères du bras robot TX60

5.7.1.4 *Mouvements en mode Point*

Dans ce mode, l'utilisateur peut amener le bras robot à un point chargé depuis une application présente dans la mémoire du contrôleur CS8C

5.7.2 *Mouvements programmés*

Il y a trois types de déplacement possibles qui assurent la transition entre les points en langage VAL3: mouvements point-à-point, en ligne droite et circulaires. A noter que dans les deux derniers déplacements, des singularités peuvent apparaître. Cela se produit lors d'un alignement de deux axes du robot, ce qui implique que le bras n'arrive pas à relier deux points, même proches.

5.7.2.1 *Mouvements point-à-point*

Seul le point de destination est important. Le robot calcule pour l'outil une courbe définie par le système de manière à optimiser la vitesse de mouvement. Le problème des singularités ne survient pas en mode point-à-point parce que c'est le système qui gère la transition dans ce type de

mouvement. En revanche, si deux axes sont alignés, il se peut que le poignet du robot effectue une rotation de 360° sur la trajectoire pour accéder au point suivant. Ce mouvement est automatique et incontrôlable, mais nécessaire pour que la destination soit atteinte sans erreur.

5.7.2.2 Mouvements en ligne droite

Comme son nom l'indique, le système calcule une trajectoire parfaitement droite entre les deux points. L'orientation est interpolée linéairement entre la position initiale et finale de l'outil.

5.7.2.3 Mouvements circulaires

Dans un mouvement circulaire, le centre outil se déplace le long d'un arc de cercle défini par trois points. L'orientation est interpolée entre ces différents points.

6 Réalisation

6.1 Principe

L'objectif du travail est de rejouer sur le bras robot TX60 les mouvements d'une main capturés à l'aide du Leap Motion Controller. La position et l'orientation de la main doivent pouvoir être reproduits soit en direct, soit en différé.

Le transfert des informations entre l'ordinateur sur lequel le Leap Motion est branché et le robot doit être le plus rapide possible. Le contrôleur CS8C du bras permet d'assurer des connexions par protocole Modbus et Ethernet. Ce dernier a été choisi car la majorité des ordinateurs d'aujourd'hui sont équipés de ports RJ45 employés le plus fréquemment pour ce protocole.

Le lien entre la main de l'utilisateur et le robot est présenté à la figure ci-dessous (Figure 21).

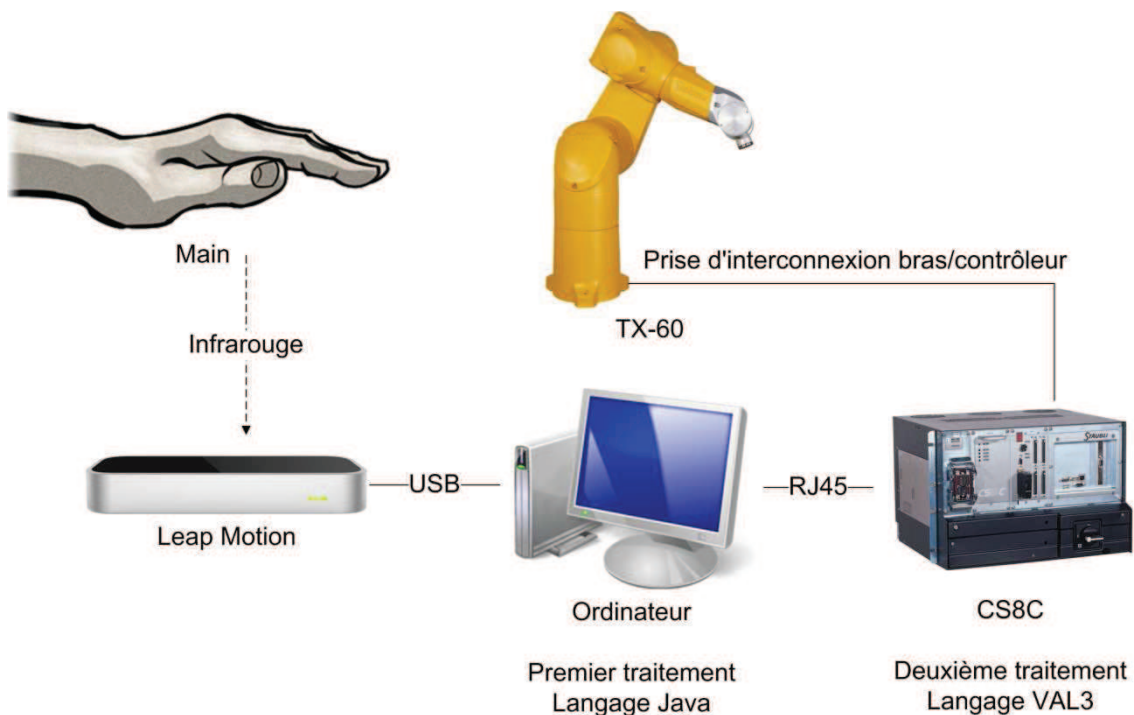


Figure 21: Transfert des informations

L'information est donc captée par le Leap Motion et interprétée par le pilote sur l'ordinateur. Ces données sont soumises à un premier traitement dans un programme en Java qui consiste à adapter les changements de référentiels par des matrices de transformation et à interpréter les gestes spécifiques qui effectuent des actions particulières. Toutes les informations relatives au prochain positionnement du robot sont alors placées dans un paquet TCP, puis envoyées au contrôleur CS8C. Celui-ci effectue un deuxième traitement sur ces données dans le langage de programmation VAL3 afin d'éviter que le bras ne subisse de collision. Un volume de contrôle est donc défini pour le robot avec des limites minimales et maximales pour chaque axe et des contraintes d'orientation. Le TX60 effectue finalement le déplacement vers sa nouvelle cible.

6.2 Fonctionnalités

6.2.1 Pince

Le robot est capable de saisir un objet au moyen d'une pince à vérin double effet. Si l'utilisateur ramène le pouce sous la paume de sa main, la pince se ferme et maintient cet état tant que le pouce est plié.

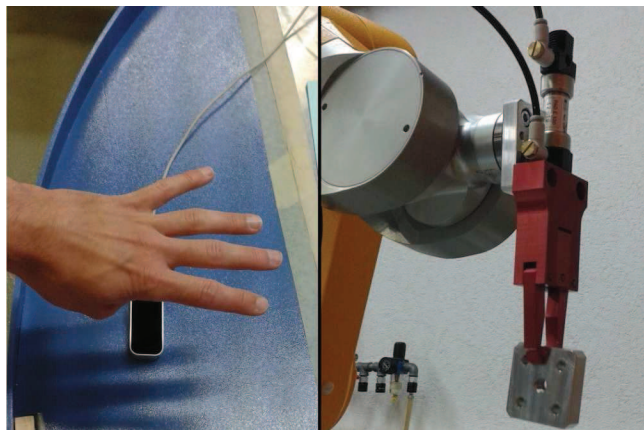


Figure 22: La pince est fermée quand le pouce est ramené sous la paume de la main

La pince pointe vers le bas par défaut, mais peut pointer vers l'avant par un double clic de l'index dans l'air (geste Key Tap).

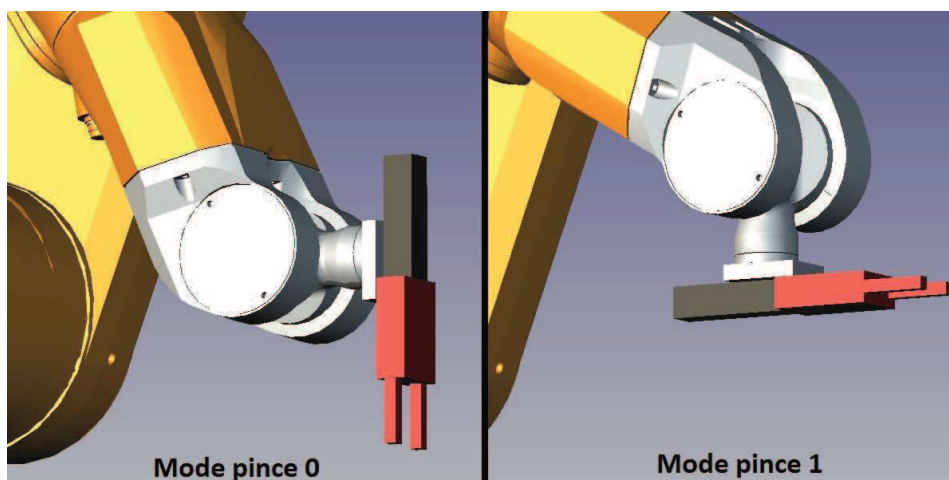


Figure 23: Modes de la pince

6.2.2 Arrêt et reprise du suivi

Le robot ne bouge pas tant que l'utilisateur garde ses quatre doigts repliés (de l'index à l'auriculaire).

6.2.3 Position relative

Il est possible, en changeant un paramètre interne au programme, de faire en sorte que la reprise du suivi de la main se fasse depuis la position où celle-ci s'ouvre. Ceci permet de ramener la main en dessus du Leap Motion, quelle que soit la position de l'extrémité du bras robot, pour éviter des instabilités de détection du périphérique.

6.3 Java

Les directives du travail imposent la lecture et l'enregistrement de mouvements. Le programme LM Capture réalisé durant le projet de semestre a donc été repris et modifié pour commander le robot. Ce nouveau programme a été nommé LM-TX60 Capture

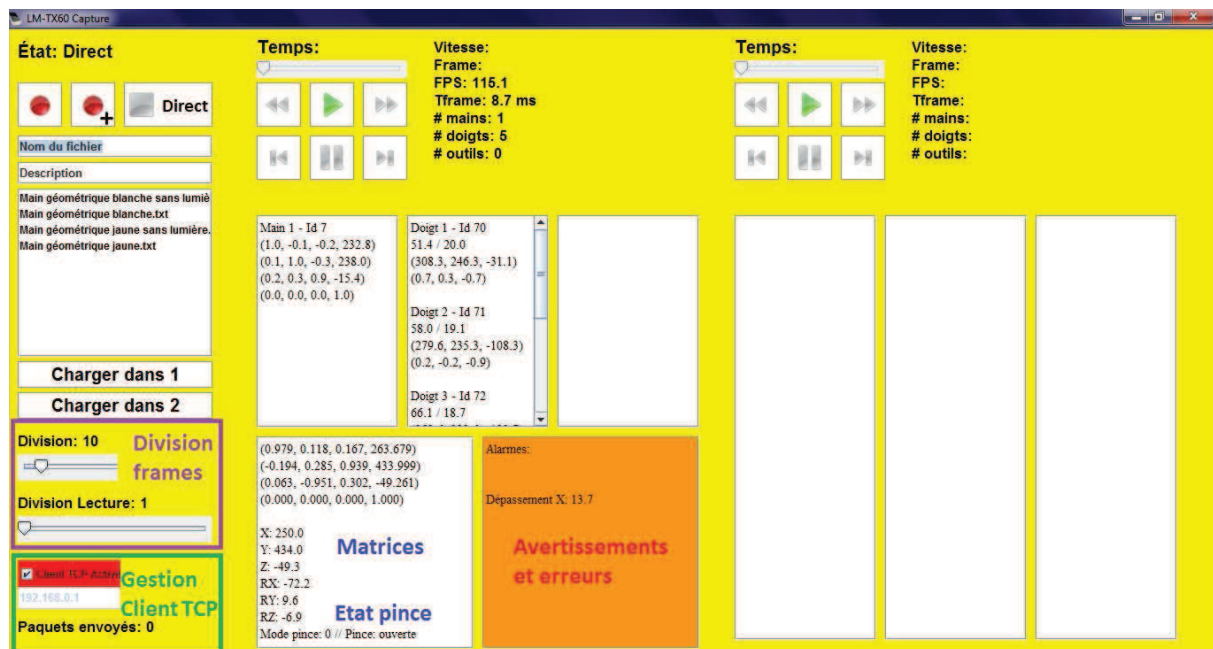


Figure 24: Troisième programme: LM-TX60 Capture

En apparence, les fenêtres des deux projets sont très similaires. Le nouveau programme propose en prime:

- Le choix de la division du nombre de données envoyées au robot: une frame chaque division sera traitée.
- L'activation du client TCP et définition de l'adresse IP à atteindre: cette dernière peut cibler le bras TX60 réel, mais elle peut aussi pointer vers un ordinateur avec le programme SRS capable de simuler le comportement du robot.
- Deux champs de texte supplémentaires pour le lecteur no 1. Le premier affiche les informations transformées et envoyées au bras robot et le deuxième alerte l'utilisateur en cas d'erreur (rouge) ou d'avertissement (orange).

6.3.1 Classes

Le projet Java possède six classes:

6.3.1.1 *MyListener*

Cette classe contient la méthode principale "main" et gère toutes les méthodes propres aux bibliothèques du Leap Motion. Les actions sous les différents modes du programme (Direct, Enregistrement et Lecture) sont définies dans cette classe. Elle effectue également tous les calculs et traitements de matrices et de données jusqu'à l'assemblage du paquet TCP.

6.3.1.2 *GUILeapMotion*

Classe regroupant les méthodes gérant l'aspect graphique de la fenêtre, le timer pour la lecture et les transitions des différents modes.

6.3.1.3 *GestionFichiers*

Cette classe contient les méthodes gérant les fichiers sur le disque dur (création et écriture).

6.3.1.4 *Matrice2D*

Classe possédant toutes les méthodes pour les calculs de matrices. Ces dernières doivent être déclarées comme des tableaux de double (tableaux de deux dimensions).

6.3.1.5 *ClientTCPTabBytes*

Cette classe ne contient qu'une méthode. Elle crée un socket vers l'adresse IP entrée dans la fenêtre et y envoie un paquet TCP contenant le tableau de bytes entré en paramètre.

6.3.1.6 *Parametres*

Cette classe ne contient aucune méthode, mais regroupe les constantes internes que le programmeur peut modifier. A noter qu'il peut être dangereux (risques de collisions pour le robot) de modifier ces valeurs sans avoir une idée des conséquences qui peuvent en découler.

6.3.2 *Matrices*

La première étape consiste à prendre en compte le changement de référentiel entre la main présentée par l'utilisateur et la "main" du robot. Chaque modification de référence implique une multiplication par une matrice 4x4 qui correspond à cette modification. Ces matrices possèdent la forme suivante:

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & u_x \\ r_{21} & r_{22} & r_{23} & u_y \\ r_{31} & r_{32} & r_{33} & u_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

u_x , u_y et u_z représentent la translation selon les trois axes entre les origines des deux référentiels. La matrice 3x3 r_{11} à r_{33} représente, quant à elle, la rotation. Lorsque deux référentiels sont confondus, la matrice de transformation qui en ressort est une matrice identité.

Il y a quatre référentiels dans le système: la main de l'utilisateur, le Leap Motion, la base du robot et la main à l'extrémité du robot.

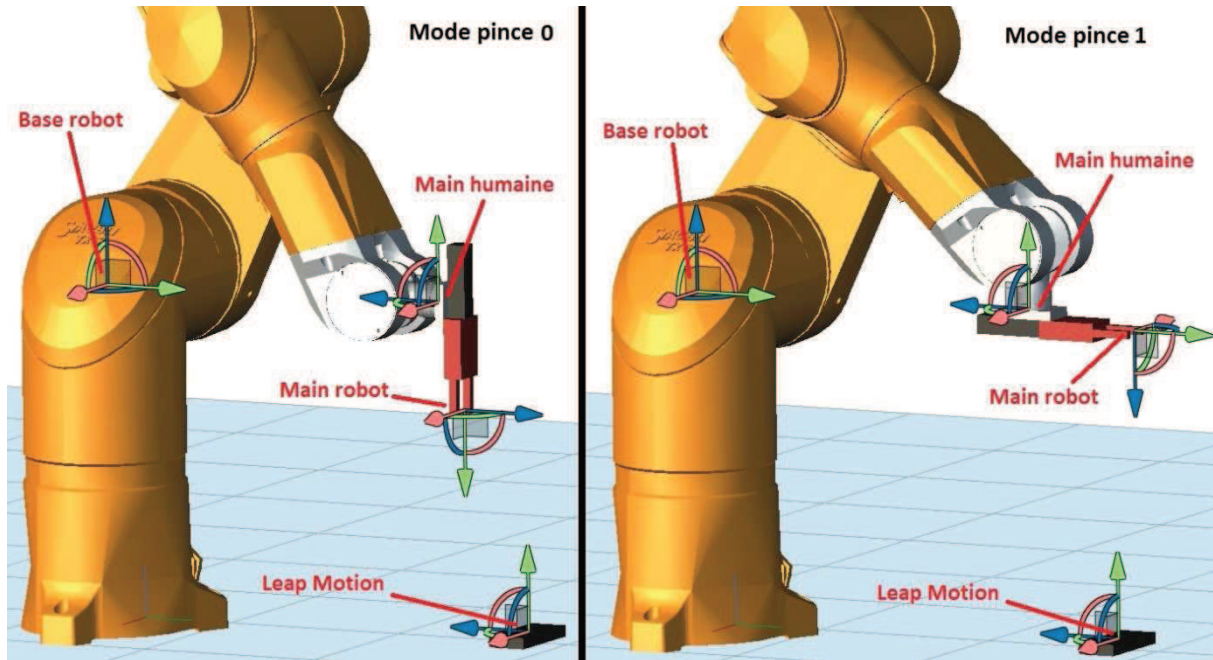


Figure 25: Référentiels pour les deux modes de la pince

Il y a donc quatre matrices de transformation à prendre en compte, une pour chaque changement de référentiel.

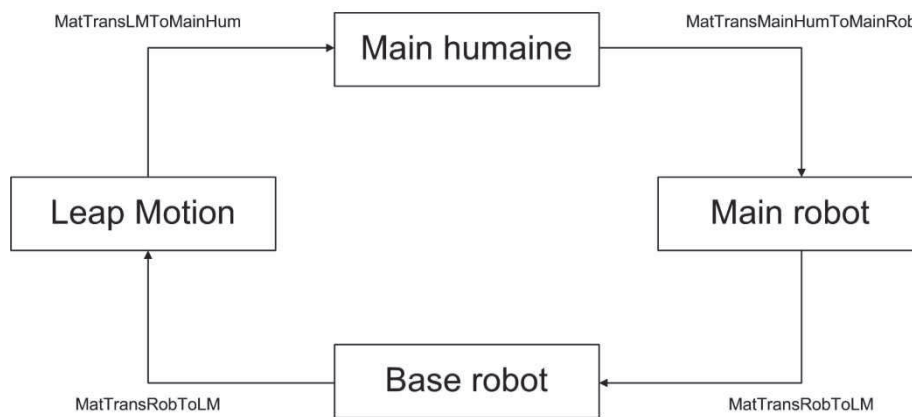


Figure 26: Matrices de transformation

6.3.2.1 *MatTransLMToMainHum*

Représente la matrice de transformation entre la main de l'utilisateur et le Leap Motion. Les composantes de cette matrice sont fournies par le périphérique à travers ses bibliothèques.

6.3.2.2 *MatTransRobToLM*

Représente la matrice de transformation entre le Leap Motion Controller et la base du robot. Dans le programme Java, cette matrice est équivalente à:

```
MatTransRobToLM = Matrice2D.Multiplication(Matrice2D.Translation(Parametres.TranslationX, Parametres.TranslationY, Parametres.TranslationZ), Matrice2D.RotationX(4, 90));
```

Il s'agit de deux transformations. Une rotation de 90° autour de l'axe X dans un premier temps, car l'orientation des deux référentiels ne sont pas identiques. Dans un second temps, on effectue une

translation paramétrable. Ces paramètres doivent correspondre à un point à l'intérieur des limites du volume de contrôle. Si la position relative est activée, les valeurs de translation changent à chaque suivi de main.

6.3.2.3 *MatTransMainHumToMainRob*

Représente la matrice de transformation entre la main de l'utilisateur et la main du robot:

```
MatTransMainHumToMainRob = Matrice2D.Multiplication(Matrice2D.RotationX(4,180),
Matrice2D.Translation(Parametres.TranslationOutilX,Parametres.TranslationOutilY,
Parametres.TranslationOutilZ));
```

Il s'agit de nouveau de deux transformations successives. D'abord une translation entre le centre de l'interface mécanique du robot (flange) et l'extrémité de la pince, suivi d'une rotation de 180° autour de l'axe X pour adapter l'orientation des deux référentiels.

Cette matrice change si le mode de la pince est modifié et que celle-ci pointe vers l'avant:

```
MatTransMainHumToMainRob = Matrice2D.Multiplication(Matrice2D.RotationX(4,-90),
Matrice2D.RotationY(4,180),
Matrice2D.Translation(Parametres.TranslationOutilX,Parametres.TranslationOutilY,
Parametres.TranslationOutilZ));
```

Ici, deux rotations suivent la translation. La première autour de l'axe Y, la seconde autour de l'axe X.

6.3.2.4 *MatTransMainRobToRob*

Représente la matrice de transformation entre la base du robot et la "main du robot" qui est l'extrémité du bras. On l'obtient par multiplication des trois matrices précédentes. (voir Figure 26)

```
MatTransRobToLM = Matrice2D.Multiplication(Matrice2D.Translation(Parametres.TranslationX,
Parametres.TranslationY, Parametres.TranslationZ), Matrice2D.RotationX(4, 90));
```

C'est cette matrice qui sera transférée au bras TX60 après traitement.

6.3.2.5 *MatStäubli*

La matrice *MatTransRobToLM* doit être adaptée au format de positionnement trsf de Stäubli. Ce format est composé de six coordonnées: trois pour la position et trois pour l'orientation {x,y,z,rx,ry,rz}.

Les trois coordonnées x, y et z correspondent à la translation depuis le point de référence. Cette dernière transformation étant appliquée avant l'orientation, ses composantes sont égales à U_x , U_y et U_z de la matrice *MatTransRobToLM*.

Les trois coordonnées rx, ry et rz correspondent aux angles des rotations qu'il faut appliquer successivement autour des axes x, y et z pour obtenir l'orientation du nouveau repère.

En multipliant trois matrices de transformation appliquant des rotations successives autour de x, y et z, on obtient ces coordonnées de rotation.

On a donc pour le format trsf:

$$x = u_x$$

$$y = u_y$$

$$z = u_z$$

$$rx = \arctan 2(-r_{23}, r_{33}) * 180/\pi$$

$$ry = \arcsin (r_{13}) * 180/\pi$$

$$rz = \arctan 2 (-r_{12}, r_{11}) * 180/\pi$$

La fonction arctan2 retourne une valeur entre $-\pi$ et π contrairement à la fonction arctan dont l'intervalle se situe entre $-\pi/2$ et $\pi/2$.

6.3.3 Vérifications et limitations

Il y a de nombreuses constantes que le programmeur peut modifier dans la classe Parametres. Il s'agit des coordonnées du point par défaut (position fixe plus ou moins centrale dans le volume de contrôle) et des limites minimales et maximales de la position pour les trois axes. Ces six limites impliquent que le centre de l'interface mécanique du robot peut se déplacer à l'intérieur d'un parallélépipède rectangle.

Le programme copie d'abord les données fournies par la frame en cours. Ces données proviennent directement des bibliothèques du Leap Motion en mode direct ou en mode enregistrement. En mode lecture en revanche, elles sont extraites du fichier txt importé. Le format txt créé en mode enregistrement a été modifié par rapport au programme LM Capture. En effet, l'état de la pince, le mode d'orientation de celle-ci et la désactivation du suivi de la main sont maintenant pris en compte dans l'enregistrement d'une trajectoire et dans sa lecture.

Une fois la frame en cours importée, les matrices sont transformées afin de prendre en compte les changements de référentiels (voir chapitre 6.3.2). Il en ressort une matrice dans le format trsf de Stäubli. L'algorithme traite alors les données de la première main de la liste. Si aucune main n'est détectée, le robot reste immobile. Si deux mains sont visibles, le robot retourne à la position par défaut. Le programme vérifie que la position cible se trouve bien dans le volume de contrôle. Si un dépassement de limite est détecté par le programme, celui-ci affiche sa valeur dans le champ de texte des alarmes.

6.3.4 Client TCP

Le transfert des données entre le programme en Java sur l'ordinateur et le programme en VAL3 sur le contrôleur CS8C du robot est assuré par un mode de communication client-serveur TCP. Les différents équipements doivent être connectés sur le même réseau. L'ordinateur depuis lequel est exécuté le programme Java joue le rôle du client. Le serveur peut être soit le bras robot, soit un autre ordinateur intégrant la simulation 3D du robot.

Le client TCP développé en Java n'a besoin que de trois informations pour pouvoir envoyer un paquet TCP valide: l'adresse IP du serveur, le numéro de port réservé pour cet échange et le tableau de bytes contenant les données traitées.

Port choisi: 1234
 Adresse IP du robot: 192.168.0.1
 Adresse IP de l'ordinateur avec simulation 3D: 192.168.0.11
 Masque de sous-réseau: 255.255.255.0

A noter que l'ordinateur depuis lequel le programme java est exécuté doit avoir le même masque de sous-réseau que le robot. La connexion risque de ne pas aboutir dans le cas contraire. L'adresse IP du client doit donc commencer par les trois mêmes valeurs (ici 192.168.0). Le dernier nombre peut être n'importe quel nombre entre 0 et 255 tant qu'il n'est pas utilisé sur le réseau. Si l'une de ces conditions n'est pas respectée en cas d'adressage automatique, il faut opérer la modification manuellement. Le port de connexion, quant à lui, doit être choisi entre 1001 et 65535. Les ports entre 0 et 1000 sont réservés.

Un socket est un point de connexion que le client ouvre avec le serveur avant de transférer les informations à envoyer. Une durée de validité avant expiration (timeout) est imposée à ce socket. Ainsi le programme est capable de détecter si le serveur a pris en compte le dernier paquet créé. Le champ d'activation du serveur apparaît en bleu si le dernier paquet a bien été reçu et en rouge si le timeout de 200ms a été dépassé.

Le paquet TCP est composé de 30 bytes:

Données	Type de donnée	Total de bytes
Position du point (6 coordonnées)	Float (4 bytes)	24
Temps jusqu'au point suivant	Float (4 bytes)	4
Mode pince (0 ou 1)	Byte	1
Etat pince (ouverte ou fermée)	Byte	1

Tableau 1: Données du paquet TCP

6.3.5 Raccourcis clavier

Certaines touches ont été liées à des fonctions:

Touche	Fonction	Depuis les modes
Esc	Quitter	Tous les modes
F1	Enregistrer	Direct - Enregistrement
F2	Enregistrer image	Direct - Enregistrement
F3	Retourner en mode direct	Enregistrement-Lecture
F4	Activer/désactiver client TCP	Tous les modes
F5	Aller à la première frame	Lecture
F6	Ralentir lecture	Lecture
F7	Aller à la frame précédente	Lecture
F8	Pause	Lecture
F9	Lecture	Lecture
F10	Aller à la frame suivante	Lecture
F11	Accélérer lecture	Lecture
F12	Aller à la dernière frame	Lecture
Page Up	Augmenter division	Tous les modes
Page Down	Diminuer division	Tous les modes
Home	Changer adresse IP cible	Tous les modes

Tableau 2: Raccourcis clavier LM-TX60 Capture

6.4 VAL3

6.4.1 Matériel

Dans SRS, les entrées et sorties pour le serveur TCP et l'électrovanne de la pince doivent être liées dans "IO Physiques" de la cellule pour être prises en compte. Un type défini existe pour chaque entrée-sortie de nature différente.

6.4.1.1 Serveur TCP

Le type SIO permet de lier une variable VAL3 à une connexion par socket Ethernet. L'objet SIO correspondant au serveur a été nommé sLMServeur. Cette variable représente donc les données brutes arrivant dans le buffer du serveur TCP. Pour que ce dernier fonctionne, il faut également configurer le serveur sur le MCP. Pour cela, on doit se rendre dans le menu "Tableau de bord" du boîtier de commande, puis ouvrir les onglets "E/S" → "Socket" → "Serveur TCP". A cet endroit, un nouveau serveur doit être créé. Plusieurs paramètres sont alors requis. Le port doit impérativement correspondre à celui du programme Java (1234 dans ce cas). Le temps entre le dernier paquet reçu et la déconnexion (Time out) doit être fixé à plusieurs minutes pour éviter que l'application ne prenne fin trop rapidement si le client est désactivé (600 secondes dans ce travail). Le caractère de fin de chaîne par défaut n'a pas besoin d'être modifié. L'algorithme de Nagle peut être activé, car cela permet de réduire le risque de saturation du réseau.

6.4.1.2 Electrovanne

Le type DIO est utilisé pour lier une variable VAL3 à une entrée-sortie numérique du système. L'objet DIO correspondant à l'électrovanne 1 a été nommé doValve1.

6.4.2 Programmes VAL3

Après le lancement de l'application, la touche "user" du MCP doit être pressée. Cette action affiche l'écran de contrôle qui permet à l'utilisateur d'interagir avec l'application.

6.4.2.1 start

Lors du lancement de l'application, c'est le programme start qui est exécuté. Il regroupe et initialise toutes les constantes utiles à l'application. Certaines de ces constantes sont les mêmes que dans le programme en Java et doivent avoir les mêmes valeurs. Après cela, le programme main est appelé.

6.4.2.2 stop

Lorsque l'arrêt de l'application est demandé, le programme stop est appelé. Au terme de son exécution, l'application se termine.

6.4.2.3 VolumeControle

Programme qui amène l'extrémité du robot dans les huit coins du parallépipède rectangle. Ce qui permet de contrôler visuellement les limites du positionnement du bras.

6.4.2.4 CalculErreur

Programme qui effectue une trajectoire traçant des lignes droites parallèles à chaque axe de référence du bras robot. Les mesures de cette trajectoire par le Leap Motion Controller seront analysées pour caractériser le périphérique.

6.4.2.5 *ChangeModePince*

Ce programme est exécuté si un changement de mode de la pince est détecté. Le robot revient alors au point par défaut, effectue la rotation nécessaire au nouveau mode, puis retourne au point précédent.

6.4.2.6 *main*

Ce programme demande d'abord à l'utilisateur si celui-ci souhaite visualiser le volume de contrôle ou tracer le chemin pour mesurer le calcul d'erreur. Il peut également activer le tracking. Dans ce cas, le bras est déplacé au point par défaut et le serveur TCP est activé. En cas d'erreur de transmission, le serveur est arrêté et une quittance est demandée sur le MCP pour relancer le tracking.

6.4.2.7 *LMServeurTCP*

Ce programme commence par attendre l'arrivée de données dans le buffer du serveur TCP. Une fois celles-ci chargées, il cherche à détecter une erreur. Cette dernière se produit si un paquet ne possède pas le bon nombre de bytes ou que le délai d'attente (Time out) est dépassé. Si aucun défaut n'est détecté, les données de sLMServeur sont enregistrées dans un tableau de bytes.

Pour que le programme Val3 puisse lire toutes les valeurs envoyées depuis Java, on utilise la commande `fromBinary()` qui permet de recomposer différentes variables à partir d'un tableau de bytes. Il faut donc indiquer en paramètres: le tableau de bytes du paquet TCP, le nombre de bytes à prendre en compte pour la valeur de sortie, le codage binaire de cette valeur et le nom de sa variable de sortie.

Le codage binaire s'écrit sous la forme d'une chaîne de caractères. Pour les nombres à virgule flottante utilisés pour les coordonnées des points et la période, il faut utiliser "4.0b". Le "4" signifie qu'une valeur est encodée sur 4 bytes (32 bits), le ".0" indique que c'est un nombre à virgule flottante et enfin "b" précise l'ordre des bytes. Pour le mode de la pince et l'état de la pince, on utilise un simple byte non signé représenté par "1".

Ces variables sont ensuite envoyées comme paramètres au programme `MovePoint`. L'utilisateur peut à tout moment mettre sur pause et relancer le tracking à l'aide du MCP.

6.4.2.8 *MovePoint*

Le programme `MovePoint` gère tous les mouvements de l'application. Quel que soit le programme qui demande un déplacement, ce dernier passe par `MovePoint`.

Les données sont traitées avec des limites cartésiennes et articulaires pour plus de sécurité. En effet, les informations arrivent en tant que type `trsf` et subissent un premier contrôle des limites de la même manière qu'en Java. Une limite de l'orientation est aussi appliquée. Ensuite, les coordonnées cartésiennes sont changées en coordonnées articulaires et un deuxième contrôle est effectué sur ces nouvelles valeurs.

La vitesse d'un point à l'autre est calculée et l'état de la pince est appliqué dans ce programme. Après cela, le buffer du robot enregistre toutes les valeurs nécessaires pour atteindre la nouvelle position et le cycle recommence.

6.5 Déplacements et fluidité du système

6.5.1 Repère

Le repère "world" a été choisi comme référentiel des positionnements. Ce repère est fixe tout au long de l'exécution de l'application.

6.5.2 Configuration

Certains points peuvent être atteints sous différentes configurations:

6.5.2.1 Epaule

La configuration de l'épaule ne modifie en rien le risque de collision. La configuration gauche a été choisie.

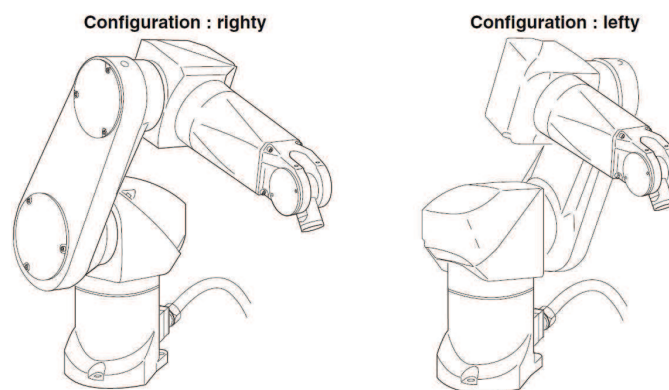


Figure 27: Configuration de l'épaule

6.5.2.2 Coude

Le coude peut être configuré vers le bas ou vers le haut. Pour éviter tout risque de collision, la configuration positive est indispensable.

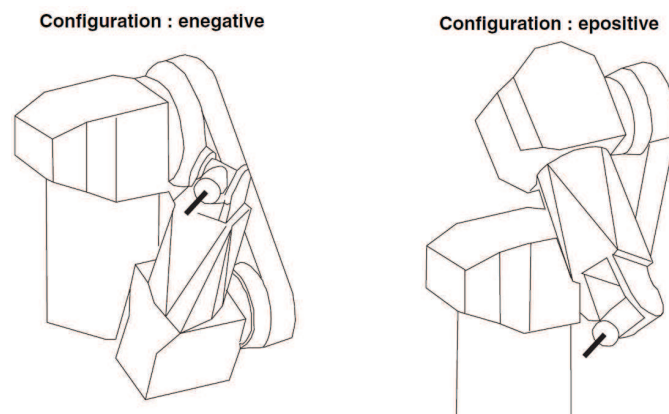


Figure 28: Configuration du coude

6.5.2.3 Poignet

La configuration positive du poignet réduit le risque de collision du vérin de la pince contre le bras.

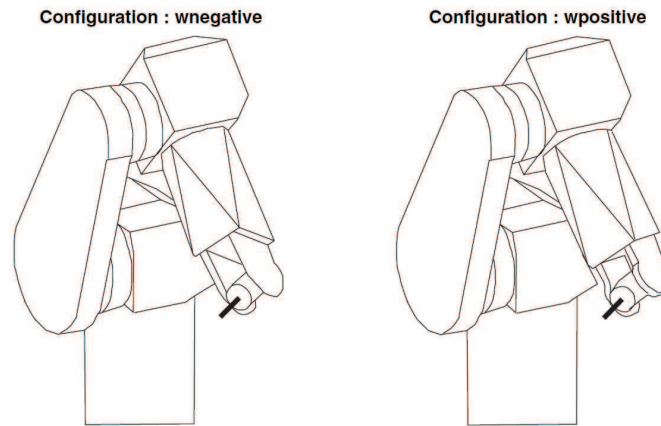


Figure 29: Configuration du poignet

6.5.3 Fluidité du mouvement

Il est possible d'influencer la fluidité du mouvement en modifiant la configuration du déplacement. Lorsque le bras se rend vers un point et qu'il reçoit dans son buffer la position du point suivant, il peut quitter la trajectoire en cours et commencer à se diriger vers la cible suivante. Le paramètre "leave" indique donc à quelle distance du point d'arrivée le robot quitte la trajectoire et le paramètre "reach" indique à quelle distance du point d'arrivée il rejoint la trajectoire suivante. Cependant, si ces valeurs sont trop élevées, le robot est moins réactif. Elles ont donc été respectivement réglées à 30 et 15. Pour activer le lissage, il faut que le paramètre "blend" soit égal à "joint".

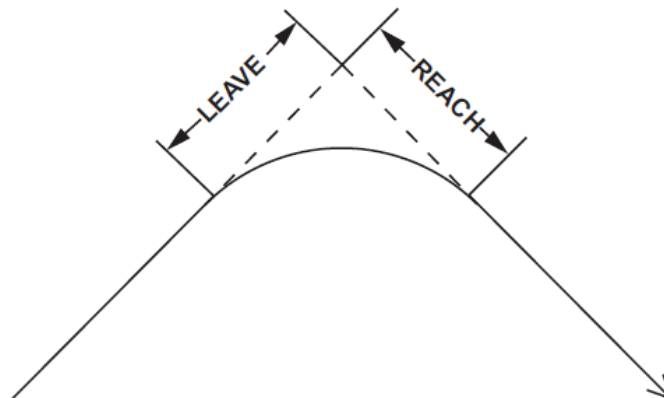


Figure 30: Leave et reach

Le robot TX60 est capable d'enregistrer plusieurs points à l'avance dans son buffer. L'exécution du code de l'application est donc en avance sur le déplacement du bras.

Deux commandes existent pour assurer une synchronisation.

La commande `waitEndMove()` permet d'attendre que la position ciblée soit atteinte avant que le programme ne passe à la suite du code VAL3. Dans ce cas, c'est comme si les valeurs `leave` et `reach` de la configuration étaient égales à 0.

La commande `resetMotion()` annule tous les mouvements enregistrés dans le buffer.

Ces deux commandes ont l'avantage d'assurer la synchronisation entre l'exécution du code et le déplacement du bras robot. Cependant, le mouvement sera saccadé. Elles ne sont donc pas utilisées en mode tracking où la fluidité est prioritaire.

6.6 Problèmes rencontrés

6.6.1 Collisions

Il ne faut à aucun moment perdre de vue la possibilité qu'une collision puisse se produire.

Comme le système réagit instantanément à la consigne donnée par la main, il serait impossible à vitesse nominale d'empêcher le choc s'il existe une quelconque erreur de paramétrage. Le mécanisme de traçage du volume de contrôle proposé au début du programme sur le MCP réduit grandement les risques: il suffit de l'exécuter à vitesse réduite pour s'assurer que rien ne se trouve dans la zone où le robot opère. Un des deux arrêts d'urgence doit toujours se trouver à proximité de l'utilisateur. Un tapis de mousse est placé sur le support du robot pour éviter que la pince ne percute un objet lorsqu'elle est dirigée vers le bas. La pince est maintenue par des vis en plastique pour encore réduire le risque.

6.6.2 Retard

Lors de la première exécution du programme sur le bras TX60, ce dernier avait énormément de retard sur le mouvement de la main de l'utilisateur, jusqu'à plus de cinq secondes. Cela était dû au fait que le Leap Motion envoyait toutes les frames qu'il capturait. Le buffer du bras robot était donc constamment saturé.

La solution à ce problème consiste à réduire le débit de données envoyé par le programme Java. Celui-ci traite donc une frame chaque division. Il existe un paramètre de division pour les modes direct et enregistrement et un autre pour le mode lecture. L'utilisateur peut modifier ces valeurs directement dans la fenêtre du programme. Lors de mon travail, l'ordinateur sur lequel est branché le périphérique reçoit environ 115 FPS. Après plusieurs tests, il s'avère que la division la plus efficace en mode direct est de 10. Le robot reçoit donc entre 11 et 12 points par secondes (115 divisé par 10). Cette division permet de ne pas saturer le buffer tout en évitant des mouvements saccadés.

La vitesse calculée à l'aide de la valeur du temps entre les deux points et de la distance qui les sépare est légèrement augmentée pour réduire le risque de retard. Au lancement de l'application, le buffer enregistre plusieurs points avant de commencer à s'y rendre. L'augmentation de vitesse permet de rattraper progressivement ce retard sans saccader le mouvement.

Malgré tous ces réglages, il reste un retard inférieur à une demi-seconde qu'il n'est pas possible d'éliminer.

6.6.3 Variation de la vitesse de lecture

Une mauvaise synchronisation du code VAL3 avec le robot peut avoir des effets négatifs en mode lecture. Le programme "LM-TX60 Capture" est capable de varier la vitesse de lecture d'un fichier txt. Si cette vitesse est ralentie, aucun problème n'en résulte. En revanche, si elle est augmentée, il y a des risques que le programme ne soit plus synchronisé avec le mouvement. Ainsi, si un enregistrement prend en compte des commutations de l'état de la pince, il est possible que ceux-ci aient de l'avance sur le mouvement du robot.

6.6.4 Limites des rotations

Il existe deux séries d'angles d'Euler pour chaque orientation. En effet, si l'on applique trois rotations de 180° par les trois axes successifs, la position ne change pas. Pour le robot, des deux façons

d'indiquer la même direction, l'une va toujours surpasser l'autre. En mode pince 0, l'orientation du point par défaut reste toujours égale à $(-90^\circ, 0^\circ, 0^\circ)$. En revanche en mode pince 1, l'orientation par défaut est égale à $(180^\circ, 0^\circ, 180^\circ)$. Lors du déplacement dans ce mode, le programme SRS change ces coordonnées en $(0^\circ, 180^\circ, 0^\circ)$, ce qui correspond au même point. Ce changement empêche de traiter correctement les valeurs limites pour ces angles. Ainsi, ce traitement est opéré en Java, car les transformations de matrices ne peuvent retourner qu'une série d'angles d'Euler par orientation. La limitation des articulations effectuée après le transfert sur le contrôleur CS8C réduit le risque de collision dû à une éventuelle erreur de transmission.

6.6.5 Client TCP

Au début du travail, le client TCP recevait en paramètres une chaîne de caractères correspondant à la concaténation de toutes les valeurs à envoyer. Certains nombres spécifiques subissaient une mauvaise transformation en type String. La valeur de sortie n'était donc pas celle souhaitée et le robot faisait un saut vers les limites du volume de contrôle. L'hypothèse la plus probable est que la transformation d'un nombre en String impliquant un caractère spécial (retour à la ligne, tabulation, etc.) peut entraîner cette erreur. Pour la corriger, le client TCP a été modifié afin de prendre en paramètres un tableau de bytes.

7 Caractérisation du Leap Motion Controller

Des mesures ont été faites pour connaître la précision du Leap Motion Controller. Le site officiel du capteur affirme que celui-ci est précis au centième de millimètres. Dans ce chapitre, les erreurs systématiques et aléatoires du périphérique seront mesurées et analysées.

Actuellement, le Leap Motion Controller est uniquement capable de détecter des mains et des pointables. Si un objet quelconque lui est présenté, il va seulement voir la main qui tient cet objet, mais ne va pas se représenter en trois dimensions l'objet tenu. Dans la plupart des cas, un objet créera des interférences; cependant, il sera parfois détecté comme une main.

En cherchant à savoir dans quelle mesure un objet ayant la forme d'une main est discerné, il s'est avéré qu'une simple forme de main découpée dans un morceau de carton est parfaitement détectée par le Leap Motion Controller.

A partir de cet élément et afin de caractériser le périphérique, une main géométrique en aluminium dont les dimensions absolues seraient connues a été créée. Il s'agit d'un cylindre aplati en lieu et place de la paume de la main et de cinq cylindres allongés de longueurs et épaisseurs différentes pour figurer les doigts. Il apparaît que le cylindre représentant la paume de la main géométrique est détecté par le Leap Motion Controller même sans aucun doigt fixé à lui. Plusieurs pièces pour maintenir fixement le Leap Motion à un emplacement précis ont aussi été fabriquées. Les plans mécaniques de ces pièces se trouvent en annexe (Annexes 2 à 11).

Il s'est d'abord avéré difficile de détecter la main géométrique. En effet, même en la plaçant bien en évidence au-dessus du Leap Motion Controller, les données étaient très instables. La main disparaissait et réapparaissait sans cesse pour le capteur. En testant plusieurs tiges de dimensions plus ou moins similaires, il est apparu que la stabilité de la détection dépend principalement du matériau de la tige. Le problème de la détection de la main découle donc du fait que l'aluminium

absorbe mieux les rayons infrarouges qu'une main réelle. Une couche de peinture blanche a alors été appliquée sur la main afin de réduire cette absorption.

Pour pouvoir effectuer un calcul d'erreur de position, il faut placer la main dans un espace en trois dimensions en connaissant sa position précise. Deux sessions de mesures ont été réalisées, la première dans le cadre du projet de semestre et la seconde lors du travail de diplôme.

Les tableaux détaillés des mesures, des références et des erreurs effectuées lors de cette première session sont joints en annexe (Annexe 12). Les scripts Matlab des deux sessions se trouvent également à la fin de rapport (Annexes 13 à 16).

7.1 Première session

7.1.1 Méthode

L'objectif de cette session est tout d'abord de calculer l'erreur de mesure de longueur, de largeur et de position de plusieurs objets dont la référence n'est pas connue. Cela implique que leur erreur est aléatoire puisque la référence est égale à la moyenne de toutes les mesures. Dans un second temps, en utilisant la main géométrique (dimensions connues), l'erreur systématique est calculée et analysée. L'angle entre les pointables est obtenu à l'aide des vecteurs des pointables. Enfin, des mesures de position sont comparées avec une référence connue.

La main géométrique est maintenue par un bras robot Mitsubishi Movemaster EX RV-M1. Ce robot est articulé avec 5 axes et possède une répétabilité de ± 0.3 mm. La version 1.2 du pilote du Leap Motion Controller est installée sur le PC. Pour effectuer les différentes mesures, le programme "LM Capture" est utilisé. Une précision au centième de millimètre ne saurait être vérifiée ici, car le bras robot Mitsubishi a une répétabilité assez importante.



Figure 31 : Bras robot Mitsubishi Movemaster et main géométrique

Quatre séries de mesures ont été réalisées. A chaque fois, l'objet à mesurer est tenu par le bras robot Mitsubishi, la frame est enregistrée avec le programme Java LM Capture, puis les calculs d'erreurs sont effectués à l'aide du logiciel Matlab version R2012a.

7.1.1.1 Crayon immobile

La première série de mesures consiste à tenir un crayon au-dessus du capteur, à une position inconnue. Le crayon est reconnu comme un outil (tool) par les bibliothèques du Leap Motion Controller. Le crayon est déplacé, puis ramené au même point entre chaque capture afin de voir si les mesures sont bien similaires.

Cinq captures sont faites dans ces conditions. A chacune d'elles, une seule frame est capturée. Les erreurs absolues et relatives sont calculées pour la longueur, la largeur et la position de l'objet. Pour tous ces calculs, la référence est la moyenne de toutes les mesures. Il s'agit donc d'une erreur aléatoire.

A l'aide des tableaux contenant les résultats, on observe que la longueur de l'objet présente une erreur relative assez importante pouvant aller jusqu'à 40 mm (47%). Les erreurs de mesure de largeur se situent entre 0.06 mm (1%) et 0.8 mm (12%).

Pour ce qui est de la position du crayon, les valeurs sont très similaires entre elles. Les erreurs sont inférieures à 0.15 mm pour l'axe des X et l'axe des Y. En revanche, l'axe Z (profondeur) est bien moins précis. L'erreur peut aller jusqu'à 0.56 mm, ce qui est considérable si on la compare aux erreurs des deux autres axes.

7.1.1.2 Main géométrique immobile

Dans cette deuxième série de mesures réalisée avec la main géométrique, le but est de calculer les erreurs de longueur et de largeur de chaque doigt en comparant les mesures avec leur référence respective, qui est connue au millimètre près. L'angle entre les vecteurs directeurs de deux doigts voisins est également comparé à sa référence qui, elle aussi, est connue. Il s'agit donc d'erreurs systématiques.

Les calculs d'erreurs de position de la main sont également réalisés. La position n'étant pas connue, il s'agit là d'erreurs aléatoires. Cinq captures sont enregistrées.

Les erreurs de mesure de longueur et largeur sont assez importantes. Celles-ci montent jusqu'à respectivement 16 mm (27%) et 1.9 mm (13%). Il s'agit ici d'erreurs systématiques puisque la référence est connue. On remarque cependant que, contrairement à ce qui a été calculé dans la série 1, les mesures sont très proches les unes des autres. Ce qui signifie que l'erreur aléatoire est très faible.

Les erreurs relatives maximales des angles entre deux doigts sont très différentes d'une mesure à l'autre. Elles apparaissent importantes pour deux des angles: 18° (30%) et 5° (18%) et faibles pour deux autres: moins de 1° (3%).

Les erreurs de position de la main sont du même ordre de grandeur que pour le crayon dans la série 1. On remarque toutefois qu'ici, c'est l'axe X (droite-gauche) qui présente l'erreur la plus importante avec 0.086 mm.

7.1.1.3 *Main géométrique en mouvement*

Cette nouvelle série de mesure a pour but de comparer les longueurs et les largeurs des doigts ainsi que les angles entre deux doigts voisins comme précédemment. Mais ici, la capture est faite pendant que la main est déplacée par le bras robot. Ceci afin de savoir si la précision varie pour un objet en mouvement par rapport à un objet immobile. Comme il y a un déplacement, il ne peut pas y avoir de calcul d'erreur de position. Toutes les erreurs calculées dans cette série sont donc systématiques, car comparées à une référence connue. A partir de l'enregistrement de la main en déplacement, 18 frames ont été récupérées.

Pour les longueurs et les largeurs des doigts géométriques, l'ordre de grandeur est plus ou moins similaire à la série précédente. Les erreurs maximales valent respectivement 22mm (32%) et 2.6 mm (20%).

En revanche, les erreurs relatives aux angles sont beaucoup plus importantes. En effet, elles peuvent monter jusqu'à plus de 22.7° (75%). La précision des informations sur les vecteurs directeurs baisse ainsi certainement si l'objet détecté est en mouvement.

Cependant, comme dans la série 2, les valeurs mesurées d'une frame à l'autre sont très similaires. L'erreur aléatoire serait donc très faible.

7.1.1.4 *Déplacement entre deux points connus*

Dans cette dernière série de mesures de la première session, la main géométrique est utilisée sans ses doigts. Le cylindre de la paume est bien reconnu comme une main par le Leap Motion Controller. L'objectif de ce dernier test est de comparer la position mesurée à deux points de référence différents dont les trois composantes spatiales sont connues.

On obtient, pour la position des deux points, des erreurs systématiques qui peuvent atteindre 8.8 mm. Pour l'axe des Y cependant, cette erreur sera toujours inférieure à 2.2 mm.

7.1.2 **Discussion**

Comme informations fournies par le Leap Motion Controller, en ce qui concerne les longueurs, les largeurs et les vecteurs directeurs de pointables, la précision n'est pas bonne. Les erreurs relatives dépassent souvent les 10%.

Pour les positions en revanche, la précision est meilleure. L'erreur aléatoire maximale sur toutes les séries de mesures vaut 0.56 mm alors que l'erreur systématique maximale s'élève à 18 mm.

D'une manière générale, on peut dire que les mesures comparées entre elles sont plutôt similaires. Mais dès que la comparaison s'opère avec une valeur de référence connue, l'écart est plus important, voire très différent. Les erreurs aléatoires sont bien inférieures aux erreurs systématiques. En appliquant une correction sur la mesure, il serait possible de réduire grandement l'erreur du capteur. De telles corrections sont effectuées dans la seconde session de la caractérisation.

7.2 **Deuxième session**

7.2.1 **Méthode**

L'objectif de cette deuxième session est de corriger les erreurs systématiques de positions en appliquant différentes méthodes de calcul pour connaître l'erreur moyenne aléatoire sur une trajectoire et l'écart type sur chaque axe.

Pour cette nouvelle série de mesures, le bras robot TX60 de Stäubli a été utilisé comme support des objets à détecter. La répétabilité de ce robot est de ± 0.02 mm. La version 2.0.3 beta du pilote du Leap Motion est installée sur l'ordinateur et le programme "LM-TX60 Capture" effectue les mesures.

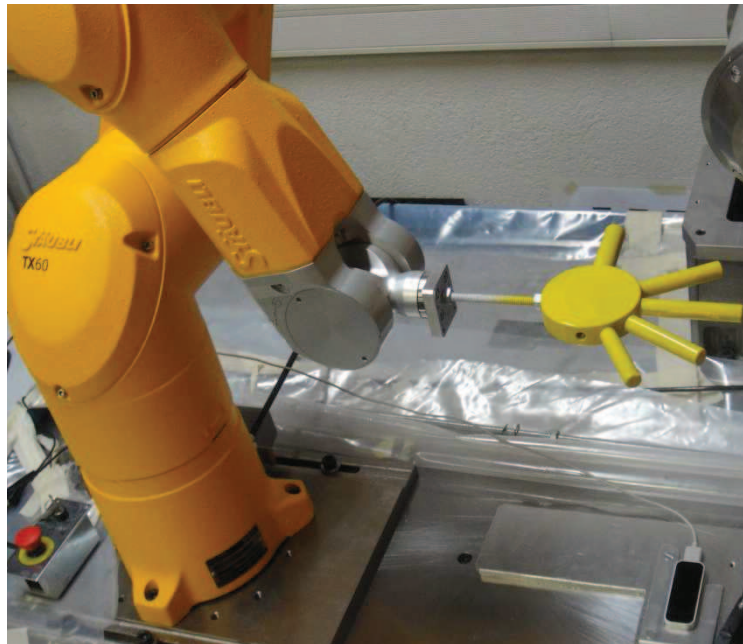


Figure 32: Bras robot Stäubli TX60 et main géométrique

Le robot est programmé en langage VAL3 pour tracer une trajectoire spécifique. Celle-ci est composée de lignes droites toujours parallèles à l'un des axes X, Y ou Z.

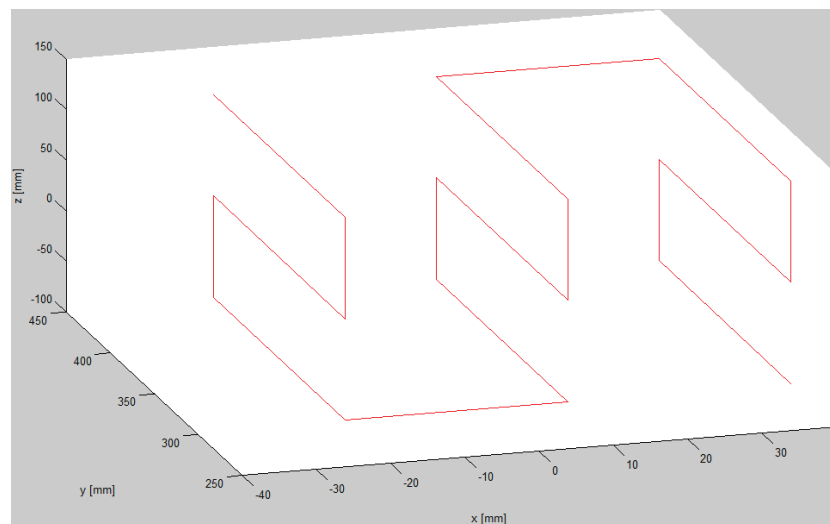


Figure 33: Forme de la trajectoire - deuxième session

A chaque capture, le robot maintient une main avec une matière différente comme surface. La main géométrique a été peinte de deux couleurs différentes: blanche, puis jaune. La main en carton sert de soutien pour celle en papier et celle en caoutchouc.

Mains maintenues par le TX60:

- Main géométrique peinte en blanc
- Main géométrique peinte en jaune
- Main en carton

-Main en papier

-Main en caoutchouc (gant)

Chaque main est capturée avec et sans lumière mais il s'avère que cette différence n'a aucune d'influence sur le résultat. La lumière située à proximité de la zone de travail est un tube fluorescent. Ces lampes émettent très peu d'infrarouges et donc, ne perturbent pas la mesure.

Deux traitements sont réalisés à l'aide du logiciel Matlab version R2012a pour chaque main. Le premier consiste à mesurer l'erreur moyenne et RMS après avoir corrigé l'erreur systématique. Cette correction est effectuée en réduisant au mieux l'écart moyen entre les points capturés et la référence. Dans le second traitement, la correction est appliquée pour un segment de chaque axe de la trajectoire et l'écart type est calculé.

7.2.1.1 Erreur moyenne

Pour ce calcul, le nuage de points de la trajectoire capturé par le Leap Motion subit une correction en translation et en rotation de manière itérative afin que la somme de tous les écarts soit finalement la plus faible possible.

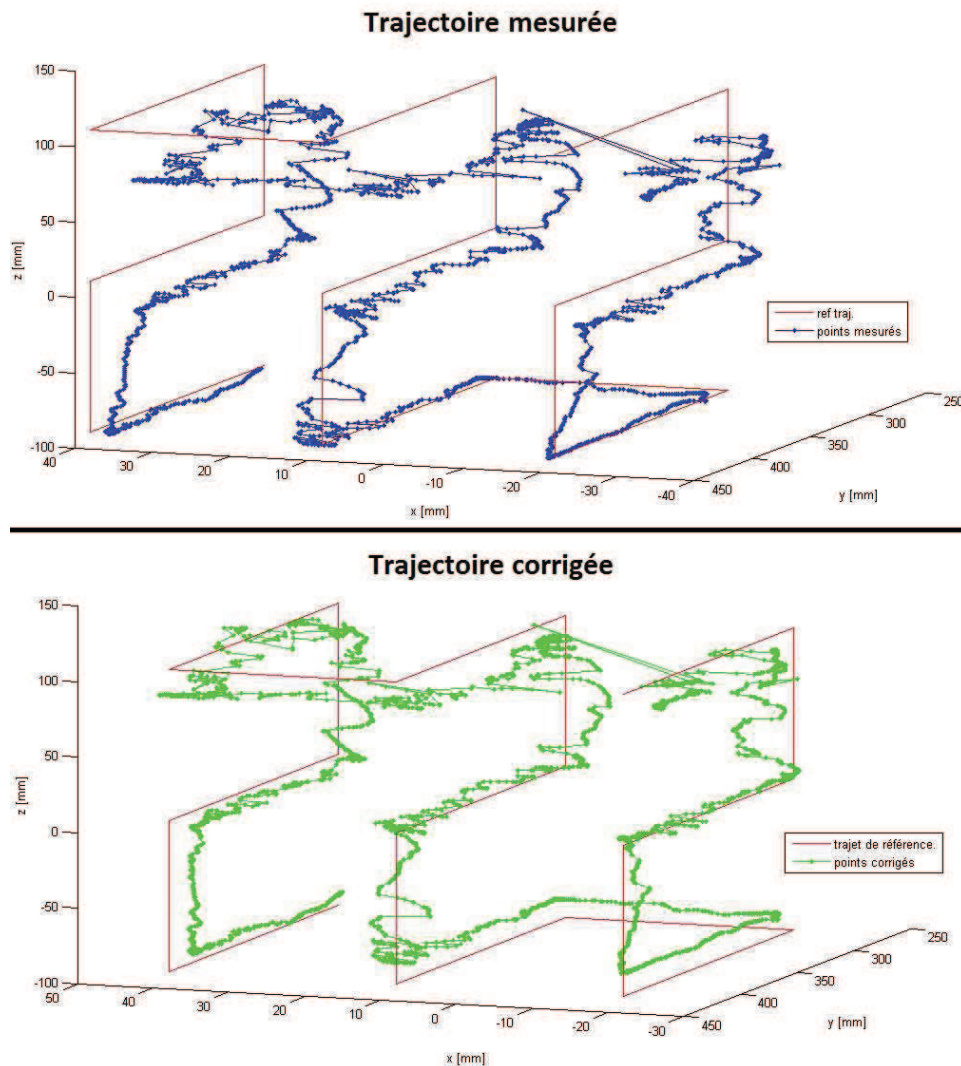


Figure 34: Correction de la trajectoire mesurée (main géométrique blanche)

Rien qu'en analysant la forme que peuvent prendre les mesures avant leur correction, on peut arriver à certaines conclusions. L'inclinaison du nuage de points ci-dessous (Figure 35) montre que même si le Leap Motion est parfaitement posé à plat, la mesure résultante peut apparaître très visiblement inclinée. Ici, en supprimant l'erreur systématique, le nuage de point est redressé.

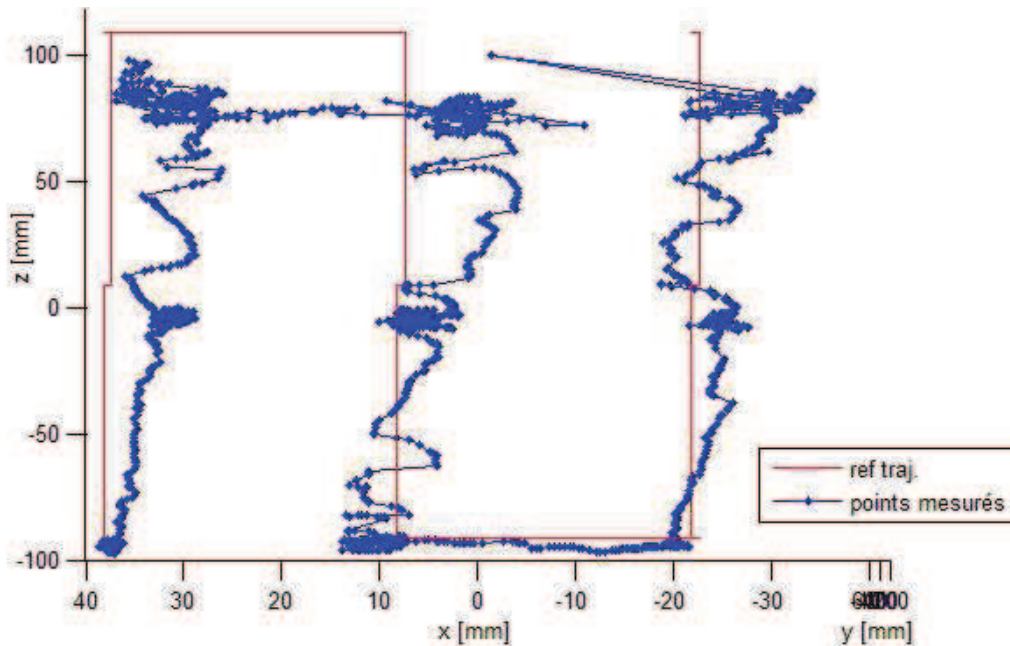


Figure 35: Exemple mesure inclinaison

En revanche, certaines erreurs ne peuvent pas être corrigées, car elles sont aléatoires. C'est le cas de la mesure suivante (Figure 36) qui montre que le capteur perd de la précision avec l'augmentation de la valeur de la coordonnées sur l'axe Z. A noter que l'axe Z dans le référentiel du robot est équivalent à l'axe Y dans le référentiel du Leap Motion.

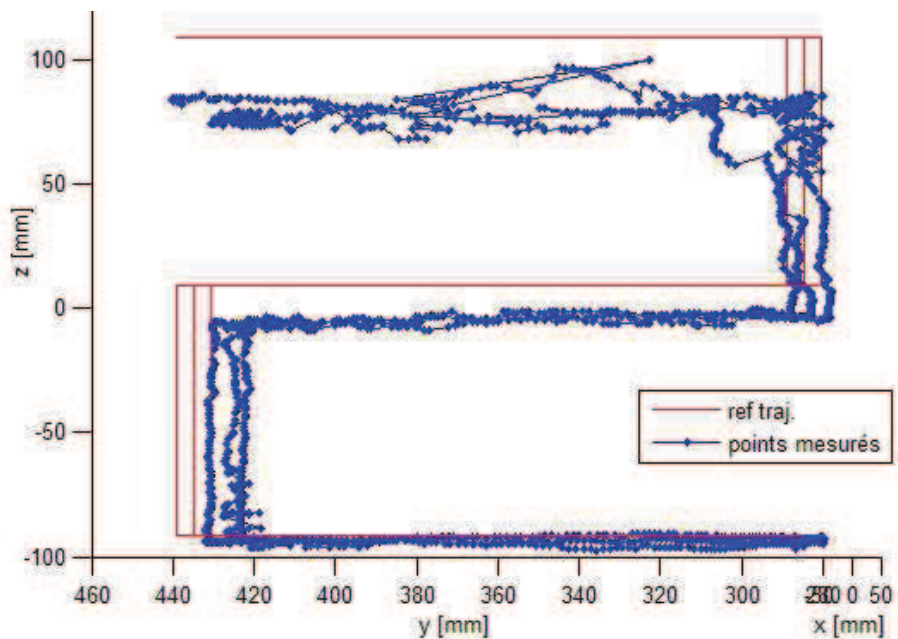


Figure 36: Exemple mesure perte de précision sur axe Z

Une fois la correction effectuée, l'erreur moyenne et l'erreur RMS sont calculées en utilisant les nouveaux écarts entre les points et la trajectoire de référence. (Tableau 3)

Valeurs en [mm]	Trajet complet	
	Erreur moyenne	Erreur RMS
Main géométrique blanche	7.0622	8.5679
Main géométrique jaune	6.8618	8.6147
Main carton	6.1118	7.5768
Main papier	6.2168	7.4008
Main caoutchouc	6.9231	8.1791

Tableau 3: Erreur moyenne et RMS pour les différentes mains

On remarque que la main en carton et celle en papier présentent des erreurs inférieures à celles d'autres matières. Avec cette méthode de correction, l'erreur systématique est grandement réduite, mais n'est pas intégralement supprimée. En effet, si l'on observe la trajectoire corrigée (Figure 34), on remarque que les points les plus bas sont au dessus de la référence correspondante et que les points les plus hauts sont au dessous. Le programme Matlab corrige la translation et la rotation, mais pour être plus efficace, il faudrait aussi chercher à corriger l'échelle. Dans l'exemple précédent (Figure 34), le signal de mesure selon l'axe Z devrait être amplifié.

7.2.1.2 Ecart type

Pour ce calcul, trois extraits de la trajectoire sont pris en considération: une pour chaque axe. Une régression linéaire est faite sur le nuage de points de telle sorte que l'erreur moyenne de la distance entre les points et la droite soit nulle. Le programme calcule alors quels sont les angles de rotation pour ramener cette droite parallèlement à l'axe de référence. Le nuage de points subit alors la même transformation.

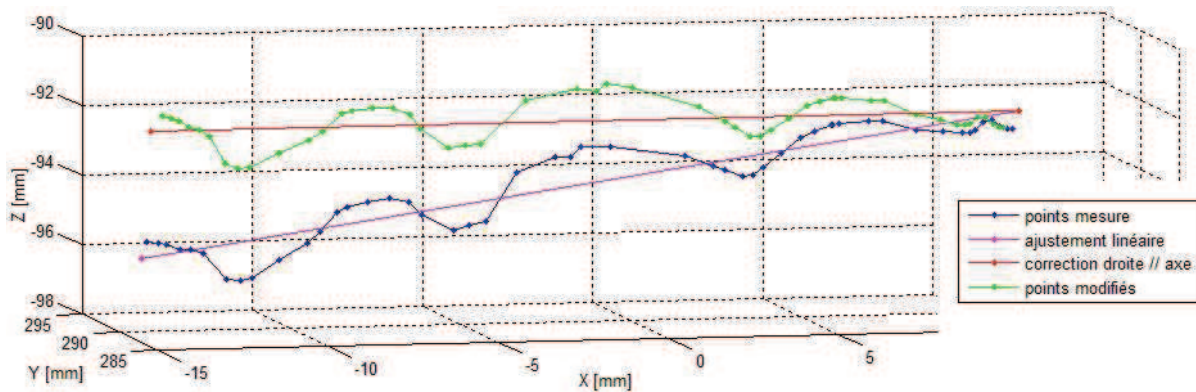


Figure 37: Régression linéaire

L'écart type est calculé (Tableau 4). Celui-ci sert à mesurer la dispersion d'un ensemble de données. Plus il est faible, plus les valeurs sont regroupées autour de la moyenne et inversement.

Valeurs en [mm]	Axe de référence: X		Axe de référence: Y		Axe de référence: Z	
	Ecart Type Y	Ecart Type Z	Ecart Type X	Ecart Type Z	Ecart Type X	Ecart Type Y
Main géométrique blanche	0.8581	0.5765	1.2157	1.0070	1.1449	1.0005
Main géométrique jaune	0.0806	0.1077	2.5503	1.2763	0.2208	0.4091
Main carton	0.3927	0.2948	1.4235	0.7193	0.1353	0.3250
Main papier	0.1095	0.2940	1.7425	0.7871	0.1726	0.4234
Main caoutchouc	0.7541	0.2590	0.5852	0.4533	0.1345	0.4867

Tableau 4: Ecart Type pour les 3 axes

L'écart maximal est également calculé (Tableau 5).

Valeurs en [mm]	Axe de référence: X		Axe de référence: Y		Axe de référence: Z	
	Ecart Max. Y	Ecart Max. Z	Ecart Max. X	Ecart Max. Z	Ecart Max. X	Ecart Max. Y
Main géométrique blanche	2.2676	1.3072	2.4125	2.1445	3.5944	2.7254
Main géométrique jaune	0.3948	0.3444	7.5191	2.7926	0.8407	0.9747
Main carton	0.8448	1.0514	2.5840	1.5098	0.3257	0.8508
Main papier	0.3654	1.0248	3.4749	1.9346	0.4740	0.9569
Main caoutchouc	2.0208	0.6422	1.6264	0.8473	0.2458	0.8995

Tableau 5: Ecart maximal pour les 3 axes

Dans ces deux tableaux, on peut supposer que l'erreur systématique a été supprimée puisque les points ont été ramenés parallèlement à l'axe de référence.

Si on compare les axes, on se rend compte que l'écart type en X est le plus élevé quel que soit l'axe de référence. Cela peut résulter du fait que les deux capteurs du périphériques sont justement alignés dans cet axe. En supposant que l'un des deux capteurs ne soit pas parfaitement aligné par rapport à l'autre et selon la manière dont sont traitées les données, la précision pourrait en être réduite.

Le code développé sous Matlab permet d'afficher une courbe de Gauss pour chaque axe. De nouveau, les mains en carton et en papier se distinguent des trois autres. Paradoxalement, les écarts type minimal et maximal reviennent tous les deux à la main géométrique jaune avec respectivement 2.55 mm et 0.081.

Les deux graphes suivants (Figure 38) présentent les courbes de Gauss des écarts type maximal et minimal.

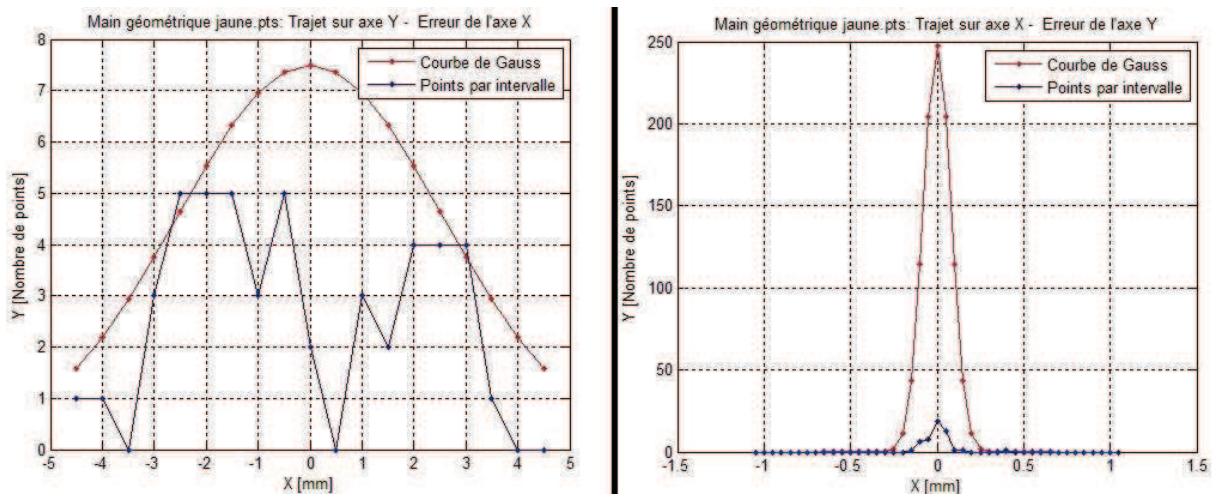


Figure 38: Courbes de Gauss - Ecart type maximal et minimal

On remarque que le point culminant de la courbe de Gauss de droite atteint des valeurs très élevées. Cela est dû au fait que sa fonction dépend de l'écart type et que justement, celui-ci est très bas.

7.2.1.3 Commandes Matlab

Toutes les figures ne peuvent pas apparaître dans ce rapport puisque les fonctions permettant leur affichage sont paramétrables. Leur nombre serait donc trop important. Les commandes Matlab ont ainsi été écrites pour pouvoir obtenir le plus simplement possible la figure voulue.

7.2.1.3.1 Commande pour erreur moyenne et affichage de la trajectoire

```
eval_traj('Main géométrique jaune sans lumière.pts', (5:2280), 'OFF');
```

Cette commande affiche les erreurs moyennes et RMS en entrant en paramètres le fichier contenant les mesures et les extrémités des points de la trajectoire. Le dernier paramètre permet d'afficher les figures de la trajectoire s'il est égal à 'ON'.

7.2.1.3.2 Commande pour écart type et affichage de la régression linéaire

```
ecartType('Main géométrique jaune.pts', (1551:1600), 'X', 'OFF');
```

Cette commande affiche l'écart type et l'écart maximal d'un nuage de points après avoir effectué une régression linéaire sur l'axe précisé en paramètres. Les autres paramètres sont identiques à la fonction précédente. La figure montre ici le détail de la régression. Cette fonction retourne le tableau des écarts entre mesures et la trajectoire de référence.

7.2.1.3.3 Commande pour courbe de Gauss

```
gauss('Main géométrique jaune.pts', MainGeometriqueJauneX, 'Y', 0.5, 3, 'OFF');
```

Cette commande prend en paramètres le nom du fichier de mesures, le tableau des écarts obtenus par la commande de l'écart type, l'axe pour lequel la courbe de Gauss doit être affichée, l'intervalle des différenciations des erreurs, la limite prise en compte et l'activation de l'affichage de la figure.

7.2.2 Discussion

Les résultats de cette deuxième session montrent que le Leap Motion Controller est un capteur bien assez précis pour faire de la télémanipulation dans passablement de domaines. Cependant, un grand nombre d'applications auraient besoin d'une précision supérieure, en particulier dans le domaine mécanique.

On remarque que la façon dont les différentes surfaces réfléchissent les infrarouges a une influence assez importante sur la précision du périphérique. Il est probable que le Leap Motion ait été optimisé pour capter les infrarouges réfléchis par la peau humaine.

La précision de l'ordre du centième de millimètre est loin d'être respectée. En donnant cette valeur, la société faisait certainement référence à la résolution, ce qui correspond à la distance minimale entre deux points entre lesquels le capteur peut faire la différence.

Dans les domaines où l'on souhaite obtenir une précision visuelle maximale, on corrige les mesures de trajectoire par des algorithmes très performants. Outre la translation, la rotation et l'amplification selon chaque axe, on prend en compte les aberrations optiques telles que les distorsions en barillet et coussinet. L'idéal pour le Leap Motion serait de pouvoir appliquer ce même type d'algorithme directement sur les mesures du périphérique afin de ne plus avoir à tenir compte de l'erreur systématique.

Une autre méthode pour obtenir d'avantage de précision dans la télémanipulation serait de réduire l'échelle entre la mesure de la main par le Leap Motion et la répétition du mouvement par le bras

robot. La précision serait augmentée du même facteur que l'échelle serait baissée. Cette alternative aurait l'avantage d'être facilement réalisable.

8 Utilisation de deux Leap Motion Controller

Comme dit précédemment, une solution pour résoudre plusieurs problèmes serait d'utiliser deux Leap Motion Controller en les plaçant à deux points de vue différents, chacun complétant le champ de vision de l'autre. Pour cela, il faut que les référentiels des deux capteurs soient adaptés avec des matrices de transformation afin que les valeurs fournies sur les positions d'un objet soient le plus proche possible. Un algorithme de correction comparant les mesures des deux capteurs pourrait effectuer ce travail.

Le problème vient du décalage des informations. En effet, il ne serait pas possible de parfaitement synchroniser les captures puisque c'est le Leap Motion qui gère la fréquence de retour des frames en fonction du type de connexion USB et de la puissance de l'ordinateur. Si le système n'a pas besoin d'être mis à jour trop fréquemment, une moyenne de l'accumulation des points des deux périphériques sur une période paramétrée ne présenterait pas ce problème de désynchronisation.

Si les deux emplacements où sont fixés les capteurs sont connus, l'erreur pourrait être grandement diminuée par le fait que la position serait une moyenne des deux mesures. De plus, en cas d'instabilité de l'un des périphériques, par exemple si une main n'était soudain plus détectée par l'un des capteurs, l'algorithme pourrait ne traiter que les données reçues par l'autre, ce qui générerait sans doute un saut de la consigne, mais non un arrêt du système.

Il se trouve que le pilote du périphérique refuse actuellement d'avoir plus d'un appareil branché sur le même système. Si on essaye de le faire, le premier contrôleur branché s'active alors que le second n'est pas pris en compte. Essayer d'instancier deux contrôleurs en Java n'y a rien changé.

Lors du projet de semestre, deux alternatives ont été envisagées pour lier deux périphériques: utiliser deux ordinateurs ou utiliser une machine virtuelle comme deuxième système. Les données pourraient alors être transmises par un mode de communication interne client-serveur TCP.

La première version du pilote a empêché le Leap Motion de fonctionner sur une machine virtuelle, mais la version 2.0 beta l'a permis. Pour des raisons de temps, la combinaison de deux Leap Motion n'a pas été testée. Le pilote est en effet sorti plusieurs semaines après le début du travail.

9 Applications

9.1 Télémanipulation avec un bras robot

Le principe de ce système peut par exemple être repris dans des situations où un être humain doit effectuer diverses manipulations dans un environnement hostile. Un biologiste pourrait agir à l'intérieur d'une salle sans passer par des chambres de décontamination.

Des manœuvres répétées avec des objets lourds sont néfastes à long terme pour l'être humain. Certains robots pouvant soulever de telles charges, il est possible aujourd'hui de diriger sans peine et de façon continue des masses importantes.

Ce système permet surtout de gagner du temps sur l'apprentissage de trajectoire aux robots. Si les déplacements ne requièrent pas une précision trop importante, au lieu de programmer tous les points du geste, il suffit d'effectuer les mouvements sur le robot tout en les enregistrant. Ainsi pour répéter le geste, il n'y a plus qu'à relire la trajectoire capturée.

9.2 Leap Motion

Plusieurs applications pratiques peuvent être imaginées au moyen de ce périphérique, en plus de celle développée dans ce travail:

Pour les professionnels qui doivent faire fonctionner un appareil ou simplement consulter des données sur un ordinateur sans pouvoir toucher l'interface, il est facile d'entrevoir un intérêt pour ce périphérique. Cela peut être le cas pour un chirurgien en pleine opération ou pour un chimiste réalisant des expériences spécifiques.

Un pompier en combinaison volant piloter un drone équipé d'une caméra pour survoler un incendie pourrait utiliser un Leap Motion Controller afin de ne pas avoir à enlever ses gants constamment. De plus, comme pour le bras robot, le guidage peut s'avérer plus intuitif qu'avec des commandes traditionnelles.

10 Conclusion

Le but de ce travail est de faire en sorte que le bras robot TX60 de Stäubli suive le mouvement d'une main en direct et qu'il soit possible de rejouer une trajectoire précédemment enregistrée. Une pince a été ajoutée au robot pour pouvoir saisir des objets.

Le capteur utilisé pour interagir avec l'utilisateur, le Leap Motion Controller, est un périphérique aux nombreuses qualités. En effet, il présente une interactivité originale, son prix de vente est relativement bas et il offre une multitude de possibilités aux développeurs. L'objet peine cependant à intéresser un large public même si lors de l'annonce de son développement, il avait grandement fait parler de lui sur la toile et dans les médias spécialisés. L'absence de norme, le manque d'habitude des utilisateurs et les instabilités fréquentes ne jouent pas en sa faveur. Avec la version 2 du pilote, l'entreprise a prouvé qu'elle ne comptait pas baisser les bras.

Plusieurs mises au point peuvent être imaginées. Un deuxième Leap Motion en parallèle augmenterait la précision. La connaissance de la position de tous les éléments du robot permettrait d'avoir plus de liberté dans les déplacements sans ajouter de risques de collision.

Le temps à disposition ne m'a pas permis de réaliser une commande pour diriger un drone, mais les objectifs du travail ont tout de même été atteints. Le système fonctionne correctement et ne présente pas d'erreur.

Les résultats de la caractérisation du Leap Motion montrent que la précision du périphérique est parfaitement acceptable pour un bon nombre d'applications. Toutefois, il est impossible d'utiliser ce capteur pour des tâches qui exigent une erreur minimale et aujourd'hui, l'industrie emploie de plus en plus d'outils qui sont d'une grande précision. Le fait que le capteur ne détecte que des objets en forme de main ou les tiges réduit encore les applications possibles. Il ne faut néanmoins pas oublier que ce périphérique a été conçu dans l'optique d'une utilisation privée plutôt qu'industrielle.

Sion, le 11 juillet 2014

Jean-Baptiste Rebord

11 Références

MARCUARD, Jean-Daniel. Rudiments de géométrie 3D appliquée en robotique : Coordonnées homogènes : Transformations de référentiels. 2014

MAÎTRE, Gilbert, MOGHADDAM BÜTZBERGER, Fariba, PRAPLAN, Charles. Laboratoire MATLAB : Introduction à Matlab. 2008

ARUN, K. S., HUANG, T. S., BLOSTEIN, S. D. Least-Squares Fitting of Two 3-D Point Sets. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1987. vol. PAMI-9, no. 5. p. 698-700

STÄUBLI. Bras - Famille TX Série 60 Manuel d'instruction. 2013

STÄUBLI. Contrôleur CS8C Manuel d'instruction. 2012

STÄUBLI. Manuel de référence VAL3 Version 6. 2008

LEAP MOTION [en ligne]. <https://www.leapmotion.com/> (consulté le 08 juillet 2014)

LEAP MOTION developer portal [en ligne]. <https://developer.leapmotion.com/> (consulté le 08 juillet 2014)

LEAP MOTION developer portal api reference version 1 java language [en ligne]. https://developer.leapmotion.com/documentation/java/api/Leap_Classes.html (consulté le 08 juillet 2014)

LEAP MOTION developer portal api reference version 2 beta java language [en ligne]. https://developer.leapmotion.com/documentation/skeletal/java/api/Leap_Classes.html (consulté le 08 juillet 2014)

WIKIPEDIA. Leap Motion. Wikipédia [en ligne]. http://en.wikipedia.org/wiki/Leap_Motion (consulté le 08 juillet 2014)

BECK, Michel. Confronté à des ventes décevantes, Leap Motion licencie 10% de ses effectifs. PCWorld [en ligne]. 25 mars 2014. <http://www.pcworld.fr/peripheriques/actualites/confronte-a-des-ventes-decevantes-leap-motion-licencie-10-de-ses-effectifs,547583,1.htm?comments=1#comments> (consulté le 08 juillet 2014)

LEAP MOTION. Setting up a project [en ligne]. 2012-2014. https://developer.leapmotion.com/documentation/java/devguide/Project_Setup.html#eclipse (consulté le 08 juillet 2014)

STÄUBLI. Robots TX60 6 axes. [en ligne]. 2014. <http://www.staubli.com/fr/robotique/robots-4-et-6-axes/robots-petits-porteurs/tx60/>.(consulté le 08 juillet 2014)

STÄUBLI. Contrôleur robot CS8C. [en ligne]. 2014. <http://www.staubli.com/fr/robotique/contrôleurs-robots/robot-controller-cs8/cs8c/>.(consulté le 08 juillet 2014)

12 Annexes

- Annexe 1: Liste des méthodes du Leap Motion Java
- Annexe 2: Plan mécanique - Support Leap Motion
- Annexe 3: Plan mécanique - Cylindre paume
- Annexe 4: Plan mécanique - Cylindre pouce
- Annexe 5: Plan mécanique - Cylindre index annulaire
- Annexe 6: Plan mécanique - Cylindre majeur
- Annexe 7: Plan mécanique - Cylindre auriculaire
- Annexe 8: Plan mécanique - Support main Stäubli
- Annexe 9: Plan mécanique - Support 2 Leap Motion
- Annexe 10: Plan mécanique - Profilé L
- Annexe 11: Plan mécanique - Support Pince
- Annexe 12: Tableaux - caractérisation - première session
- Annexe 13: Script Matlab - caractérisation - première session - calcul erreur
- Annexe 14: Script Matlab - caractérisation - deuxième session - erreur moyenne
- Annexe 15: Script Matlab - caractérisation - deuxième session - écart type
- Annexe 16: Script Matlab - caractérisation - deuxième session - Gauss

Les codes des programmes Java et VAL3 ne sont pas annexés à ce rapport par souci d'économie de papier. Ils sont toutefois disponibles auprès de M. Jean-Daniel Marcuard (mad@hevs.ch).

Remerciements

Je remercie tout d'abord mon professeur, M. Jean-Daniel Marcuard, de m'avoir accompagné tout au long de ce travail. Un merci également à M. Christophe Truffer, M. Gilbert Maître et M. Anthony Gaspoz pour l'aide précieuse qu'ils m'ont fournie. Enfin, je remercie toute l'équipe de l'atelier mécanique de l'école pour leur disponibilité.

Cette liste présente les principales méthodes utilisées et testées (Versions 1.0 et 2.0 du pilote)

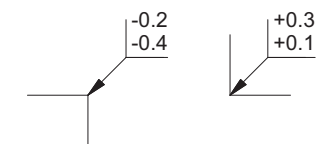
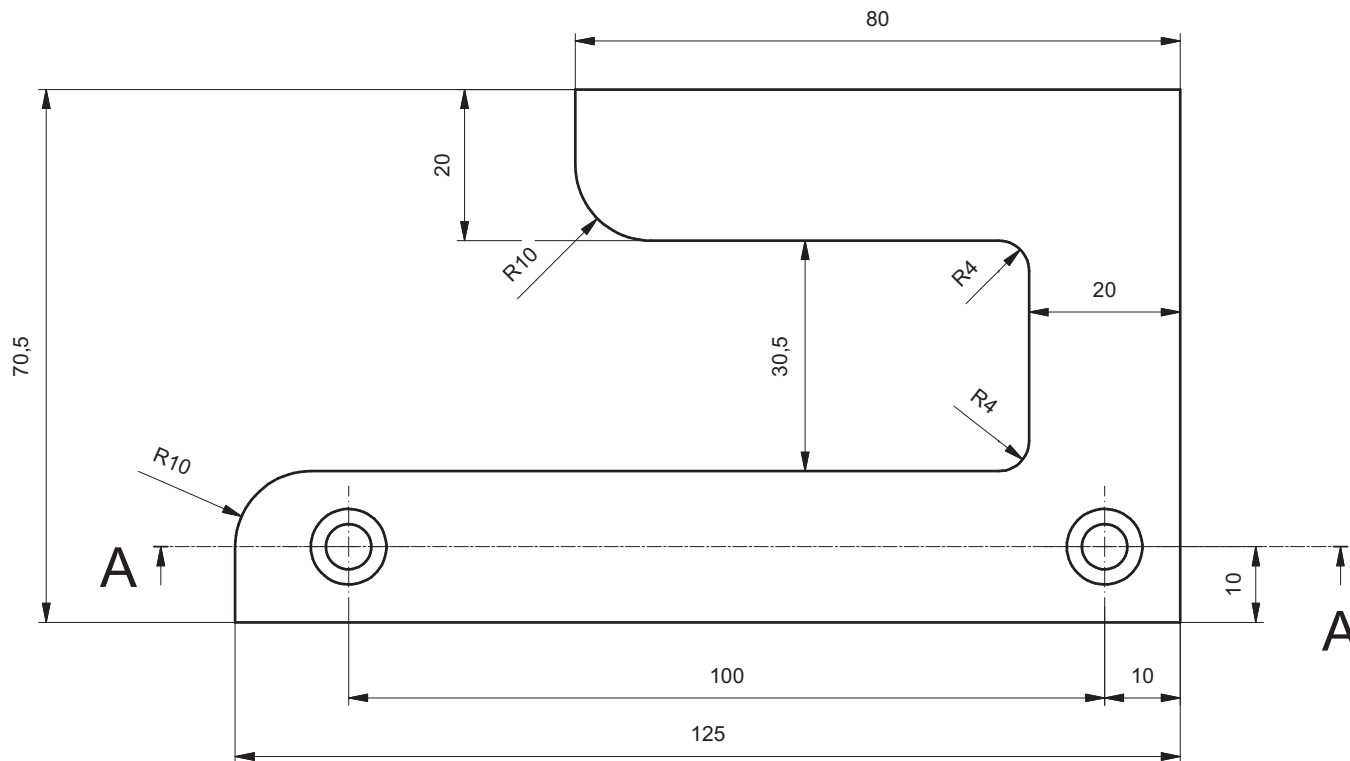
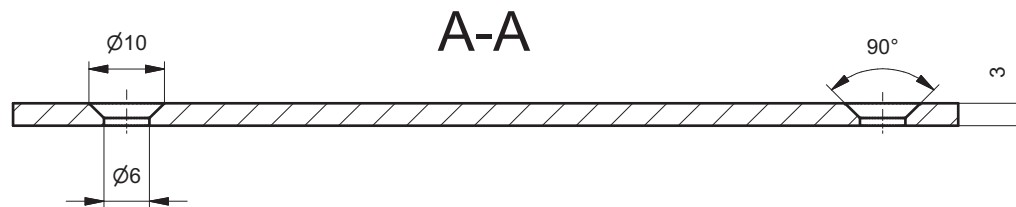
Frames par seconde	<code>frame.currentFramesPerSecond();</code>
Nombre d'objets	<code>frame.hands().count();</code> Nombre de pointables: <code>frame.pointables().count();</code> Nombre de doigts: <code>frame.fingers().count();</code> Nombre d'outils: <code>frame.tools().count();</code> Nombre de gestes <code>frame.gestures().count();</code>
Nouvel objet	<code>Hand hand = frame.hands().get();</code> La valeur entrée en paramètre dans la méthode <code>get()</code> précise à quelle main de <code>HandList</code> se rapporte la main <code>hand</code> Exemple: <code>Hand hand = frame.hands().get(0);</code> → main 0 de <code>HandList</code> Pointables: <code>Pointable pointable=frame.pointables().get();</code> Doigts: <code>Finger finger = frame.fingers().get();</code> Outils: <code>Tool tool = frame.tools().get();</code> Gestes: <code>Gesture gesture = frame.gestures().get();</code>
Id	<code>hand.id();</code> Pointables: <code>pointable.id();</code> Doigts: <code>finger.id();</code> Outils: <code>tool.id();</code> Gestes: <code>gesture.id();</code>
Méthodes propres aux mains	
Retourne la main placée le plus à gauche	<code>frame.hands().leftmost();</code> Main placée le plus à droite: <code>frame.hands().rightmost();</code> Main placée le plus au centre: <code>frame.hands().frontmost();</code>
Position du centre de la paume de la main	<code>hand.palmPosition();</code>
Vecteur normal à la paume de la main	<code>hand.palmNormal();</code>
Vecteur égal à la direction montrée par la main	<code>hand.direction();</code> Ce vecteur est toujours perpendiculaire au vecteur normal de la paume de la main. Il est possible d'obtenir le vecteur perpendiculaire aux deux vecteurs de la main par produit vectoriel afin d'avoir toutes les composantes d'une matrice de transformation

Vecteur vitesse		<code>hand.palmVelocity();</code>													
Main gauche		<code>hand.get(0).isLeft();</code>	Retourne true si la main est une main gauche et false dans le cas contraire												
Pince		<code>frame.hands().get(0).pinchStrength();</code>	Retourne une valeur entre 0 et 1 -0: les extrémités du pouce et de l'index sont écartées -1: les extrémités du pouce et de l'index se touchent												
Saisie		<code>frame.hands().get(0).grabStrength();</code>	Retourne une valeur entre 0 et 1 -0: les doigts sont tendus -1: les doigts sont repliés												
Doigt 0 rattachée à la main hand		<code>Finger finger = hand.fingers().get(0);</code>													
			<table border="1"> <thead> <tr> <th>Numéro du doigt sur la main</th> <th>Doigt</th> </tr> </thead> <tbody> <tr> <td><code>hand.fingers().get(0)</code></td> <td>Pouce</td> </tr> <tr> <td><code>hand.fingers().get(1)</code></td> <td>Index</td> </tr> <tr> <td><code>hand.fingers().get(2)</code></td> <td>Majeur</td> </tr> <tr> <td><code>hand.fingers().get(3)</code></td> <td>Annulaire</td> </tr> <tr> <td><code>hand.fingers().get(4)</code></td> <td>Auriculaire</td> </tr> </tbody> </table>	Numéro du doigt sur la main	Doigt	<code>hand.fingers().get(0)</code>	Pouce	<code>hand.fingers().get(1)</code>	Index	<code>hand.fingers().get(2)</code>	Majeur	<code>hand.fingers().get(3)</code>	Annulaire	<code>hand.fingers().get(4)</code>	Auriculaire
	Numéro du doigt sur la main	Doigt													
	<code>hand.fingers().get(0)</code>	Pouce													
	<code>hand.fingers().get(1)</code>	Index													
	<code>hand.fingers().get(2)</code>	Majeur													
<code>hand.fingers().get(3)</code>	Annulaire														
<code>hand.fingers().get(4)</code>	Auriculaire														
Méthodes propres aux pointables (même chose pour les doigts et outils)															
Position du pointable		<code>pointable.tipPosition();</code>													
Vecteur directeur du pointable		<code>pointable.direction();</code>													
Longueur du pointable		<code>pointable.length();</code>													
Epaisseur du pointable		<code>pointable.width();</code>													
Vitesse du pointable		<code>pointable.tipVelocity();</code>													
Pointable tendu		<code>Pointable.isExtended();</code>	Retourne true si le pointable est tendu et false dans le cas contraire												
Gestes															

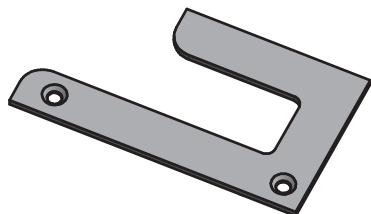
<p>Activation d'un geste</p>	<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> <code>controller.enableGesture();</code> </div> <p>L'activation des gestes doit être faite dans la méthode onConnect Le paramètre de la fonction correspond au geste que l'on veut détecter:</p> <table border="1" data-bbox="624 349 1396 539"> <thead> <tr> <th><i>Paramètre</i></th> <th><i>Geste</i></th> </tr> </thead> <tbody> <tr> <td>Gesture.Type.TYPE_CIRCLE</td> <td>Circle</td> </tr> <tr> <td>Gesture.Type.TYPE_SWIPE</td> <td>Swipe</td> </tr> <tr> <td>Gesture.Type.TYPE_KEY_TAP</td> <td>Key Tap</td> </tr> <tr> <td>Gesture.Type.TYPE_SCREEN_TAP</td> <td>Screen Tap</td> </tr> </tbody> </table>	<i>Paramètre</i>	<i>Geste</i>	Gesture.Type.TYPE_CIRCLE	Circle	Gesture.Type.TYPE_SWIPE	Swipe	Gesture.Type.TYPE_KEY_TAP	Key Tap	Gesture.Type.TYPE_SCREEN_TAP	Screen Tap
<i>Paramètre</i>	<i>Geste</i>										
Gesture.Type.TYPE_CIRCLE	Circle										
Gesture.Type.TYPE_SWIPE	Swipe										
Gesture.Type.TYPE_KEY_TAP	Key Tap										
Gesture.Type.TYPE_SCREEN_TAP	Screen Tap										
<p>Durée du geste</p>	<div style="border: 1px solid black; padding: 5px;"> <code>gesture.durationSeconds();</code> </div>										

P001

Ra 3,2



Tol. générales ISO 2768-mK



1	1	Aluminium	Support Leap Motion			
Pos.	Quantité	Matière	Dénomination / Caractéristiques			
Pos.	Menge	Werkstoff	Benennung / Merkmale			
PrS: Application d'un Leap Motion à l'apprentissage de trajectoires			Dessiné Gezeichnet	JB Rebord	12.03.2014	Echelle Massstab 1:1
			Contrôlé Geprüft			

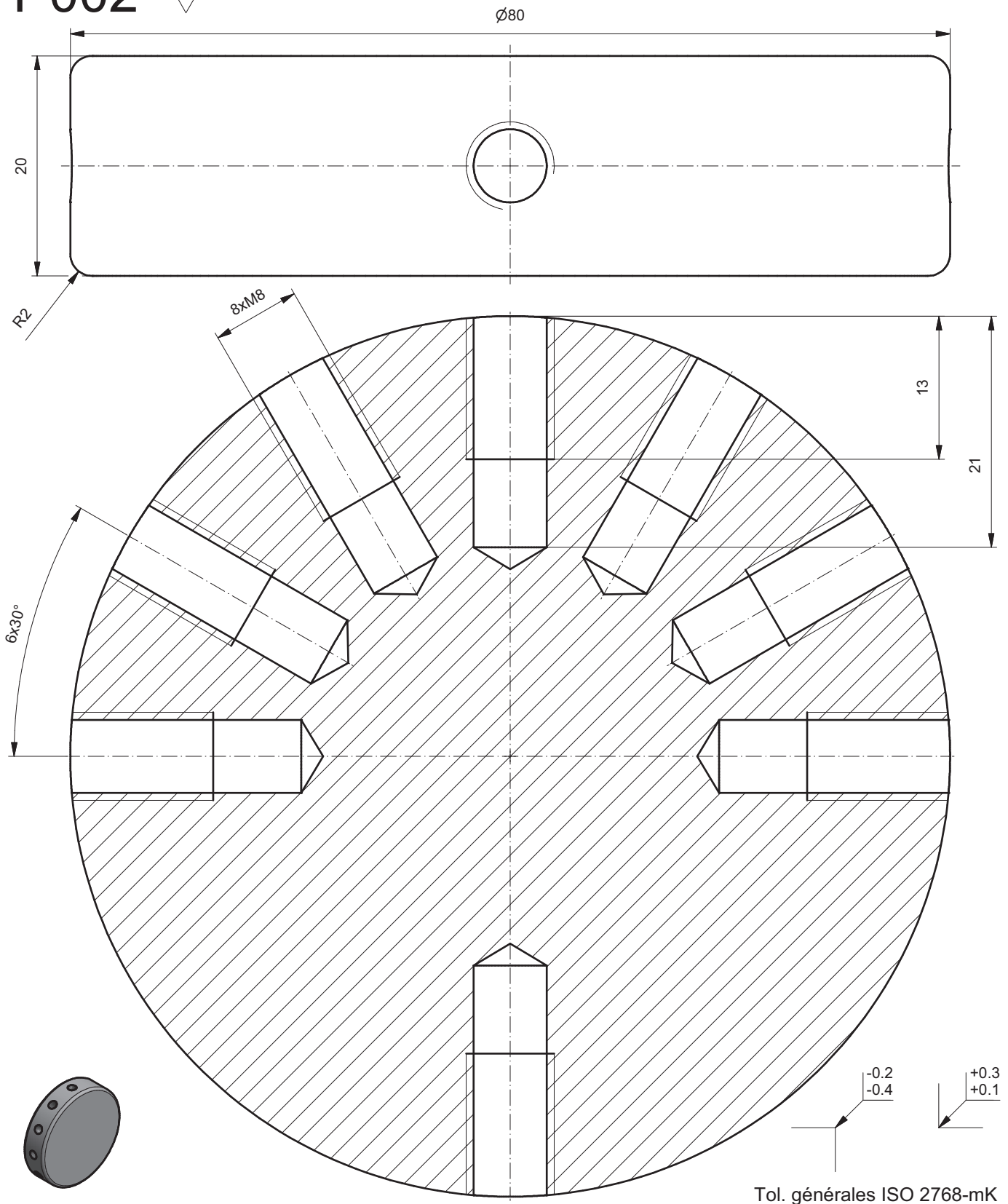
Fichier C:\Users\JB\Documents\Inventor\PrS2014LeapMotion\PrS\P001_SupportLeapMotion.idw
Datei

Hes·so VALAIS WALLIS

Annexe 2

P002

Ra 3,2

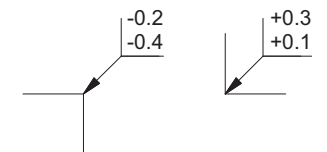
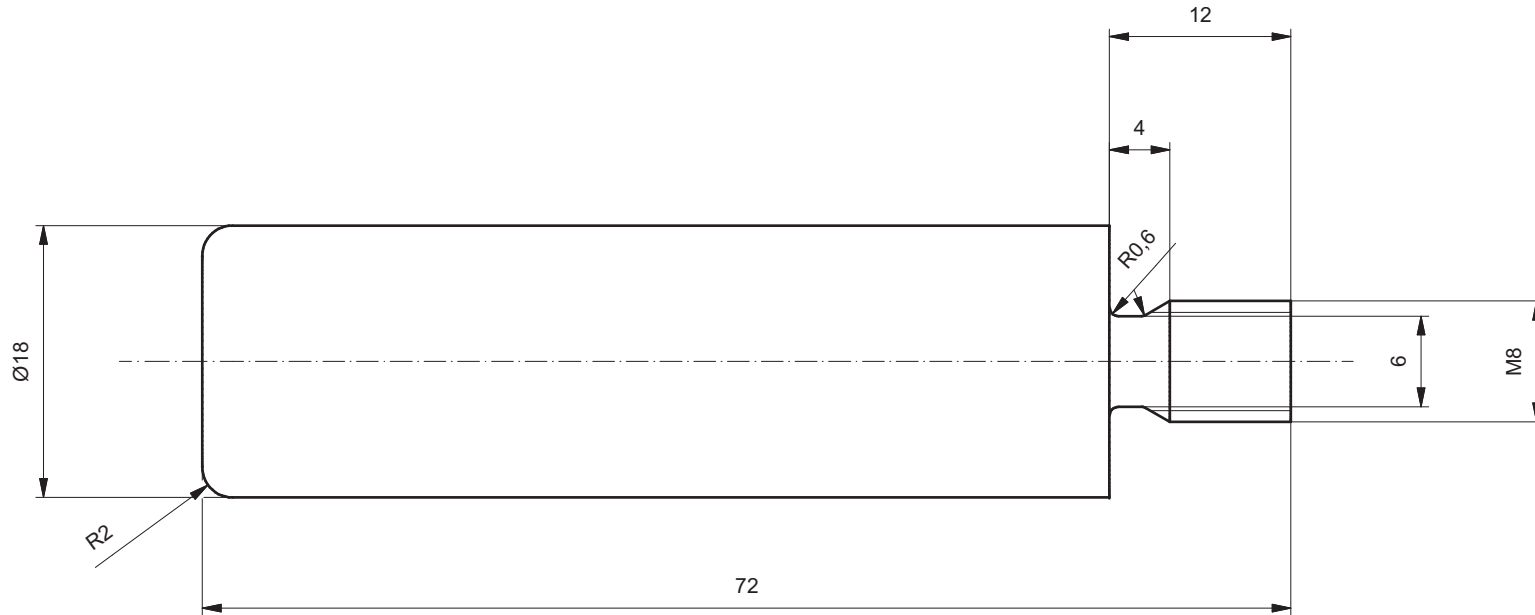


Tol. générales ISO 2768-mK

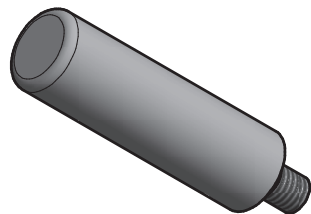
1	Aluminium	Cylindre Paume		
Pos.	Quantité	Matière	Dénomination / Caractéristiques	
Pos.	Menge	Werkstoff	Benennung / Merkmale	
PrS: Application d'un Leap Motion à l'apprentissage de trajectoires		Dessiné	JB Rebord	12.03.2014
		Gezeichnet		
		Contrôlé		Echelle
		Geprüft		2:1
Fichier	C:\Users\JB\Documents\Inventor\PrS2014LeapMotion\PrS\P002_CylindrePaume.idw			
Datei				
			Annexe 3	

P003

Ra 3,2



Tol. générales ISO 2768-mK

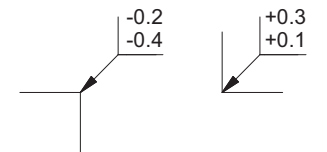
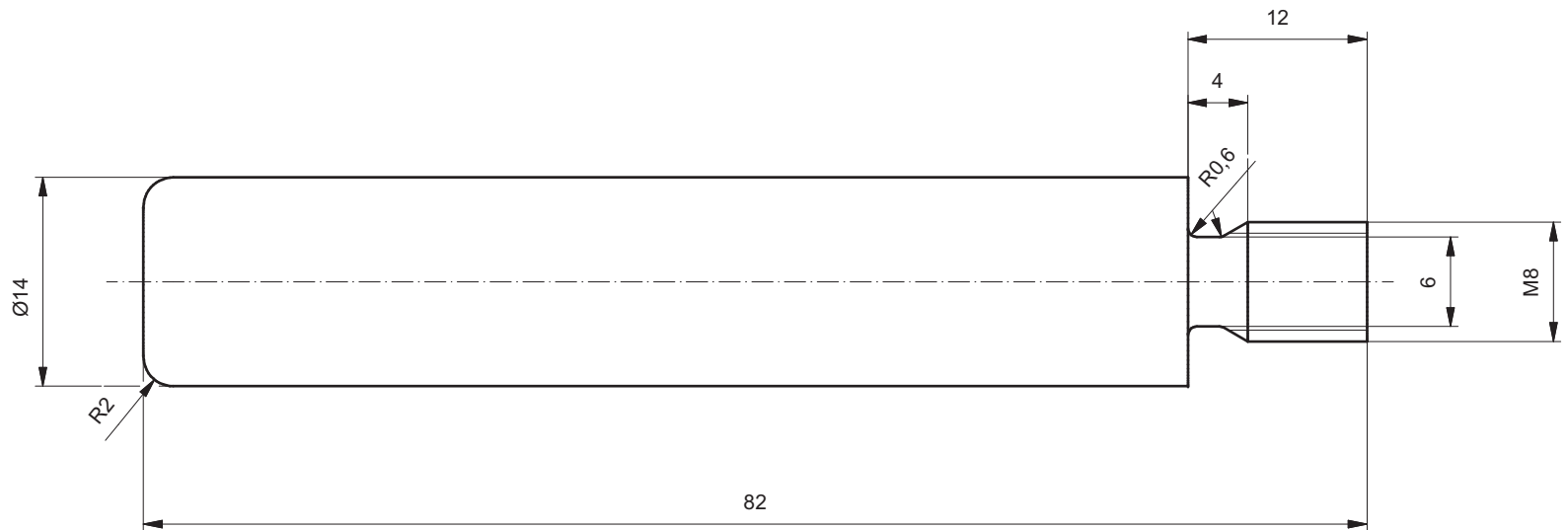


Pos.	1	Aluminium	Cylindre Pouce		
Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale		
PrS: Application d'un Leap Motion à l'apprentissage de trajectoires			Dessiné Gezeichnet	JB Rebord	12.03.2014
			Contrôlé Geprüft		
Fichier Datei	C:\Users\JB\Documents\Inventor\PrS2014LeapMotion\PrS\P003_CylindrePouce.idw				

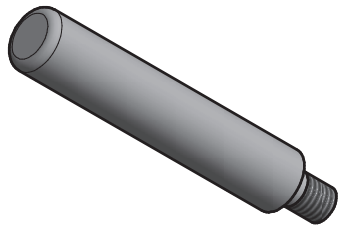
Echelle
Massstab
2:1

P004

Ra 3,2



Tol. générales ISO 2768-mK

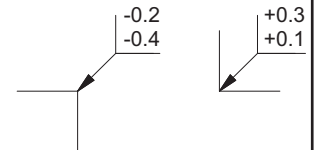
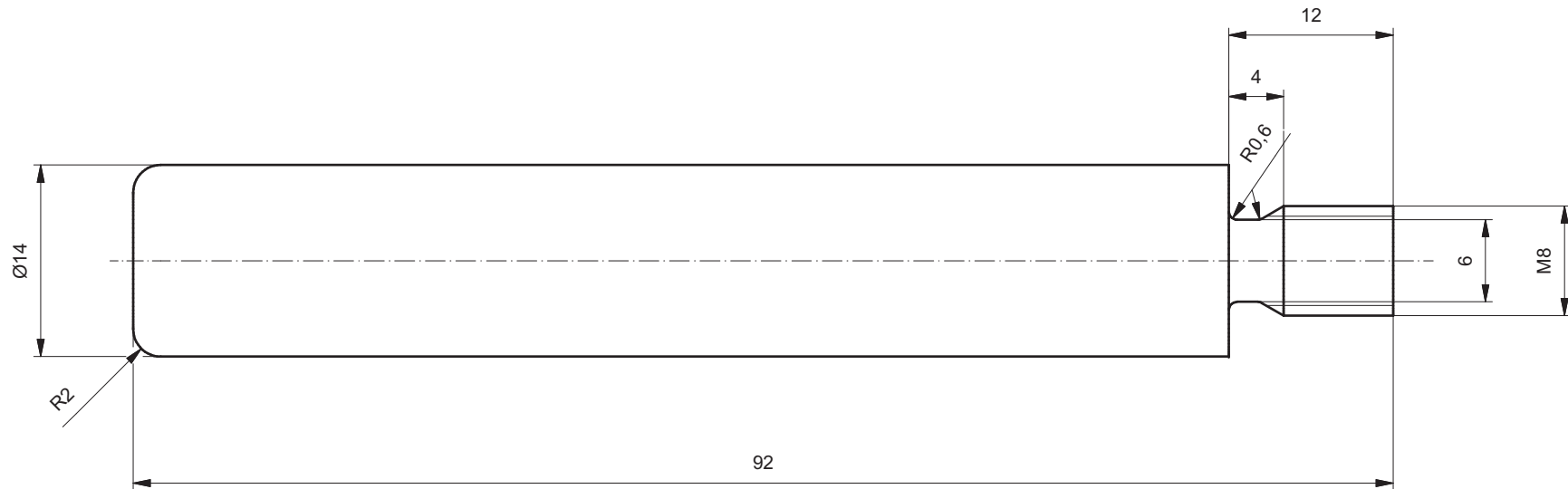


	2	Aluminium	Cylindre Index - Annulaire		
Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale		
PrS: Application d'un Leap Motion à l'apprentissage de trajectoires			Dessiné Gezeichnet	JB Rebord	12.03.2014
			Contrôlé Geprüft		
Fichier Datei			C:\Users\JB\Documents\Inventor\PrS2014LeapMotion\PrS\P004_CylindreIndexAnnulaire.idw		

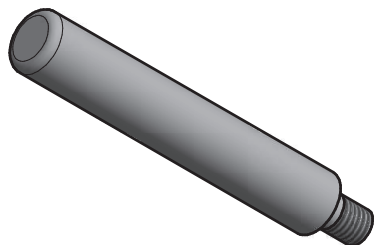
Echelle
Massstab
2:1

P005

Ra 3,2



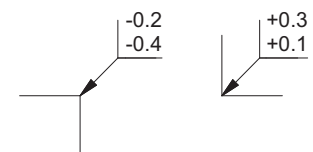
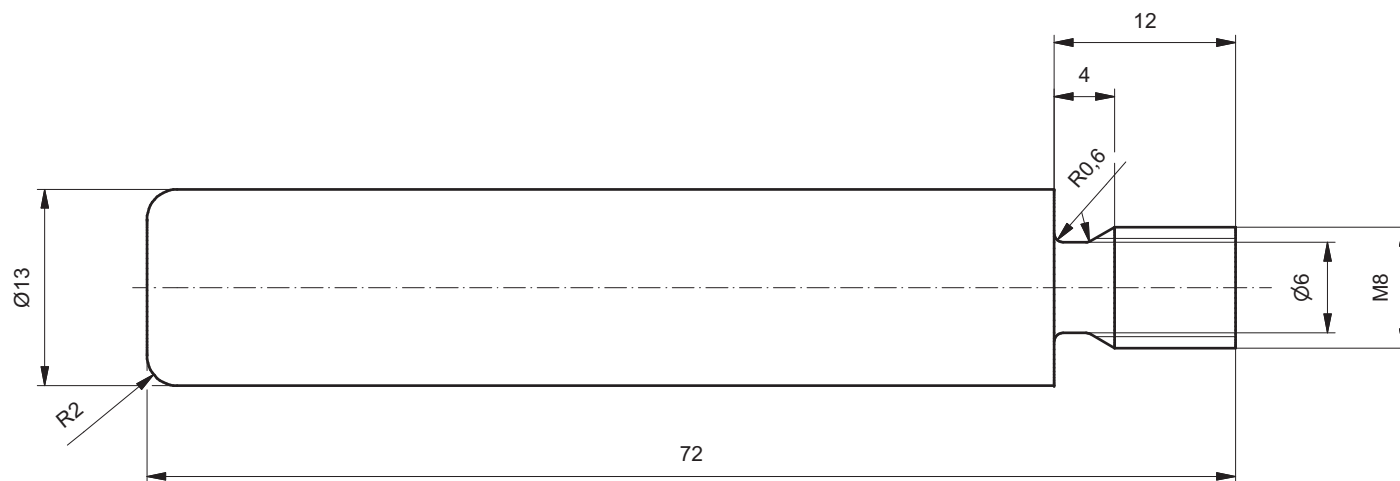
Tol. générales ISO 2768-mK



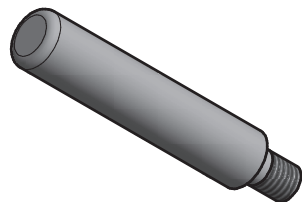
Pos.	1	Aluminium	Cylindre Majeur				
Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale				
PrS: Application d'un Leap Motion à l'apprentissage de trajectoires				Dessiné Gezeichnet	JB Rebord	12.03.2014	Echelle Massstab
				Contrôlé Geprüft			2:1
Fichier Datei	C:\Users\JB\Documents\Inventor\PrS2014LeapMotion\PrS\P005_CylindreMajeur.idw						

P006

Ra 3,2

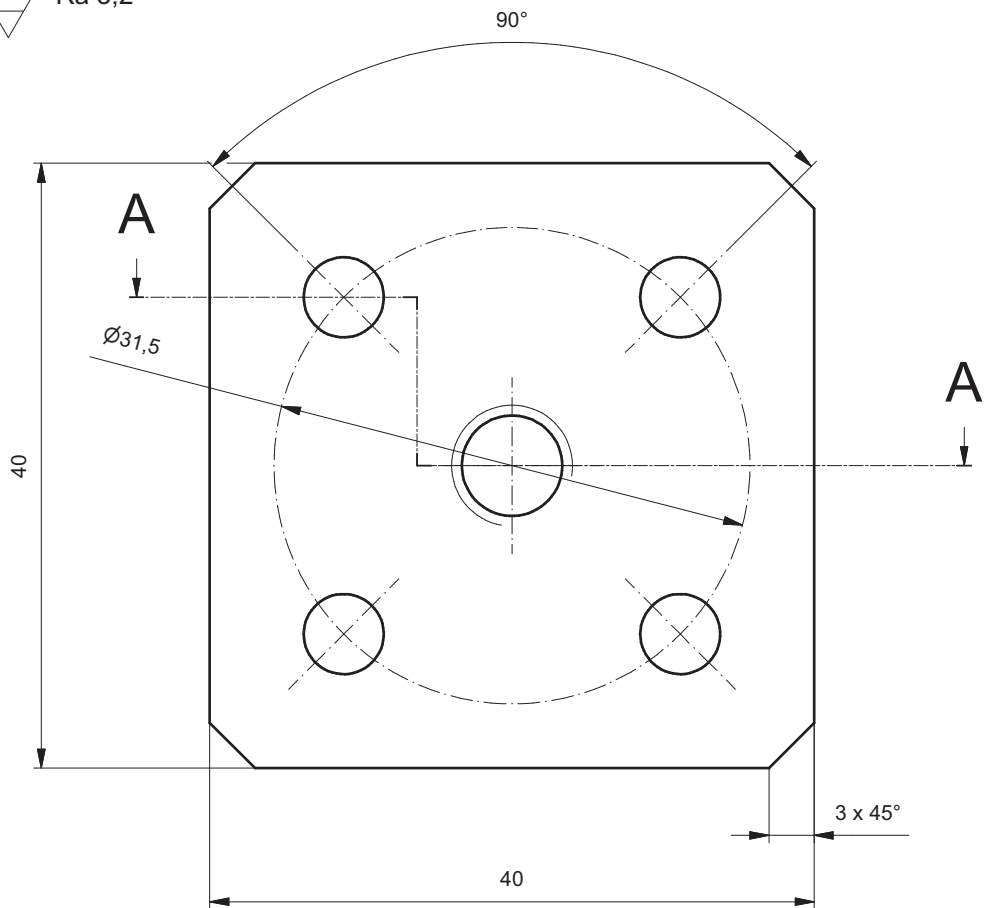


Tol. générales ISO 2768-mK

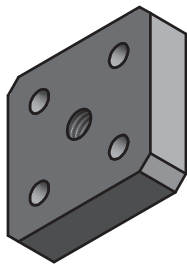
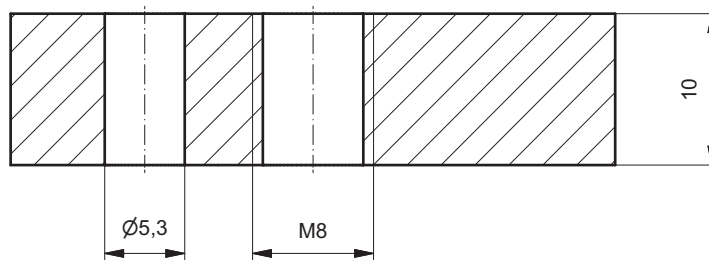


	1	Aluminium	Cylindre Auriculaire			
Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale			
PrS: Application d'un Leap Motion à l'apprentissage de trajectoires			Dessiné Gezeichnet	JB Rebord	12.03.2014	Echelle Massstab 2:1
			Contrôlé Geprüft			
Fichier Datei	C:\Users\JB\Documents\Inventor\PrS2014LeapMotion\PrS\P006_CylindreAuriculaire.idw					
			Annexe 7			

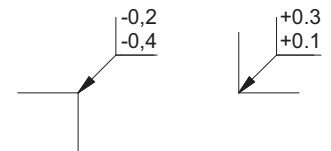
P001 $\sqrt{\text{Ra } 3,2}$



A-A



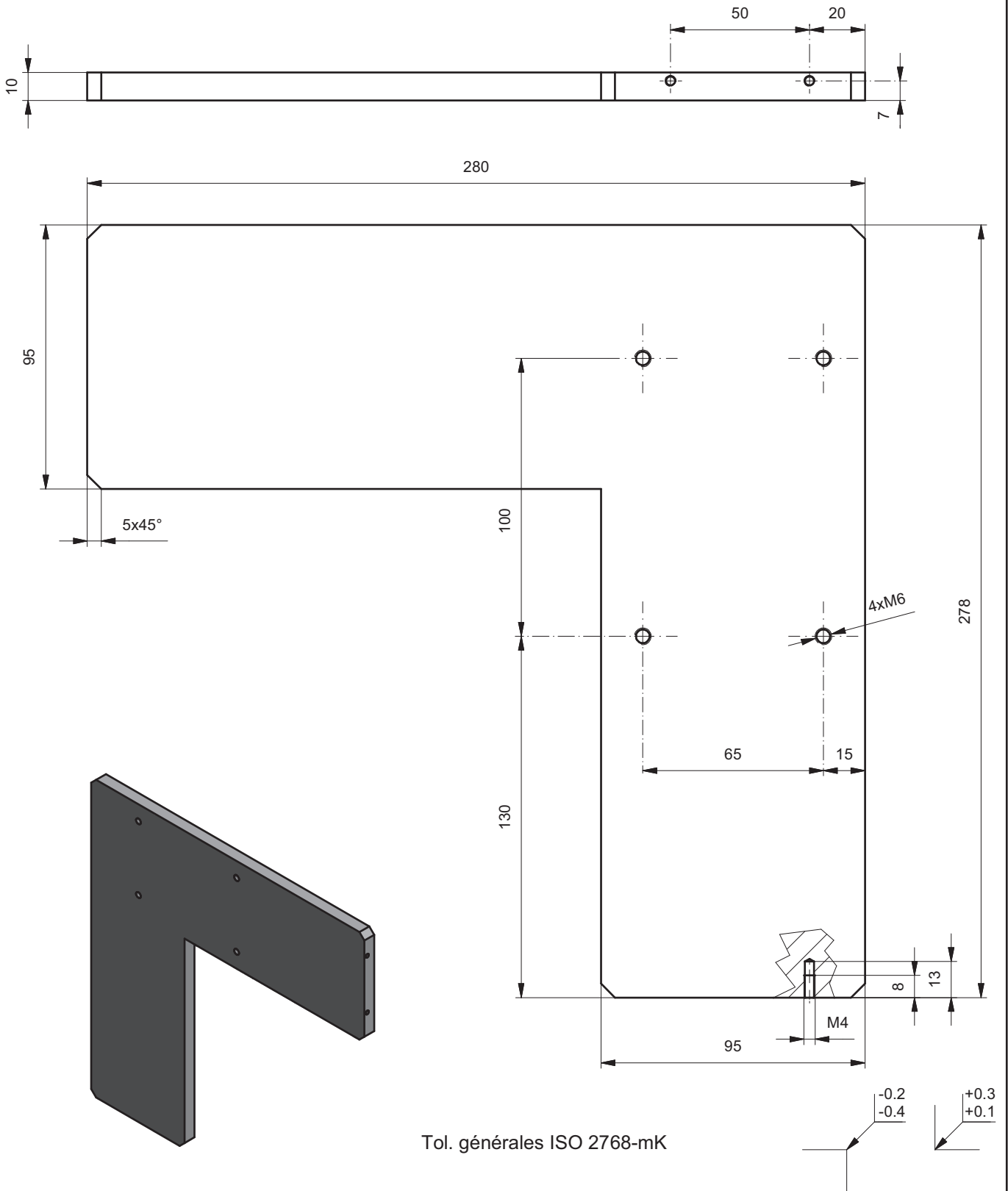
Tol. générales ISO 2768-mK



1	1	Aluminium	Support Main Stäubli			
Pos.	Quantité	Matière	Dénomination / Caractéristiques			
Pos.	Menge	Werkstoff	Benennung / Merkmale			
TD: Application d'un Leap Motion à l'apprentissage de trajectoires			Dessiné Gezeichnet	JB Rebord	23.05.2014	Echelle Massstab 2:1
			Contrôlé Geprüft			

Fichier C:\Users\JB\Documents\Inventor\TD2014LeapMotion\P001_Support_Main_Staubli.idw
Datei

P002 $\sqrt{\text{Ra } 3,2}$



2	1	Aluminium	Support LM
Pos.	Quantité	Matière	Dénomination / Caractéristiques
Pos.	Menge	Werkstoff	Benennung / Merkmale

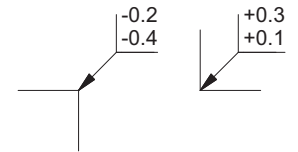
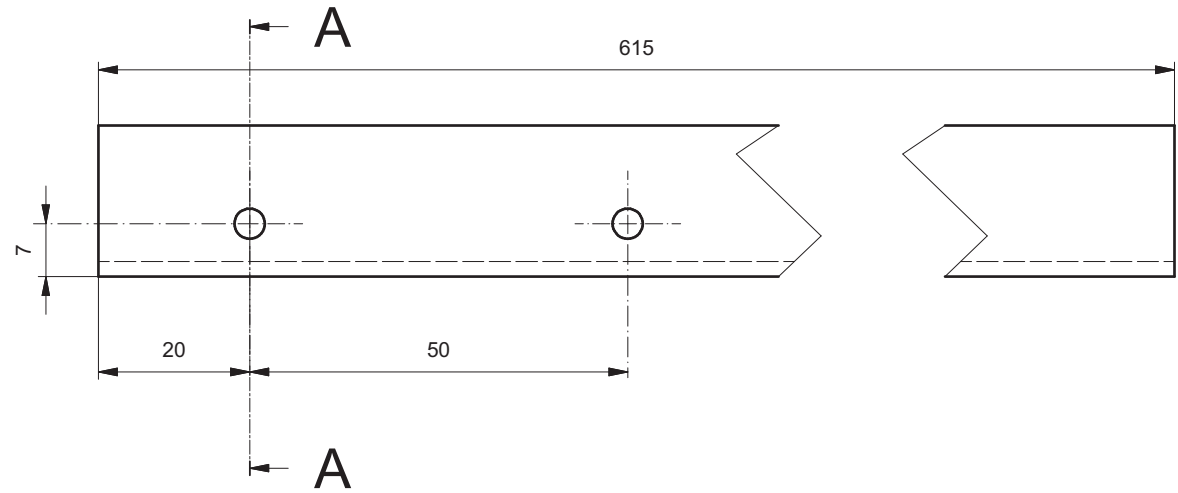
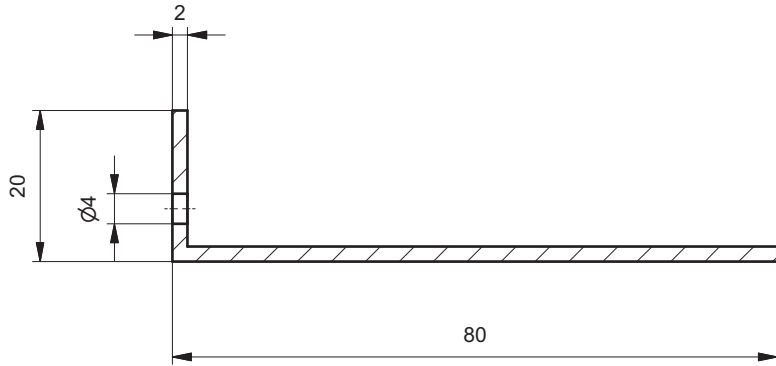
TD: Application d'un Leap Motion à l'apprentissage de trajectoires	Dessiné Gezeichnet	JB Rebord	23.05.2014	Echelle Massstab 1:2
	Contrôlé Geprüft			

Fichier Datei C:\Users\JB\Documents\Inventor\TD2014LeapMotion\P002_Support_LM.idw

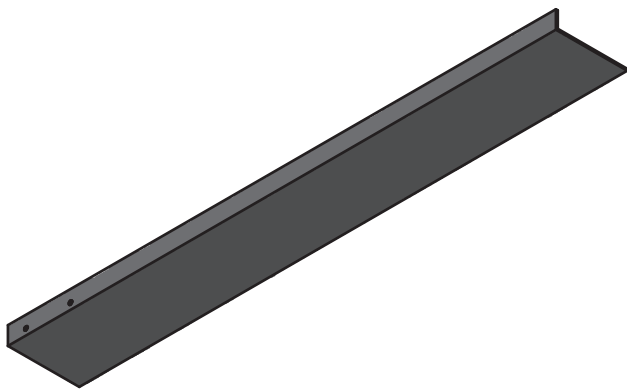
P003

Ra 3,2

A-A



Tol. générales ISO 2768-mK



3	1	Aluminium	Profilé L			
Pos.	Quantité	Matière	Dénomination / Caractéristiques			
Pos.	Menge	Werkstoff	Benennung / Merkmale			
TD: Application d'un Leap Motion à l'apprentissage de trajectoires			Dessiné Gezeichnet	JB Rebord	23.05.2014	Echelle Massstab 1:1
			Contrôlé Geprüft			

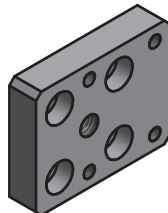
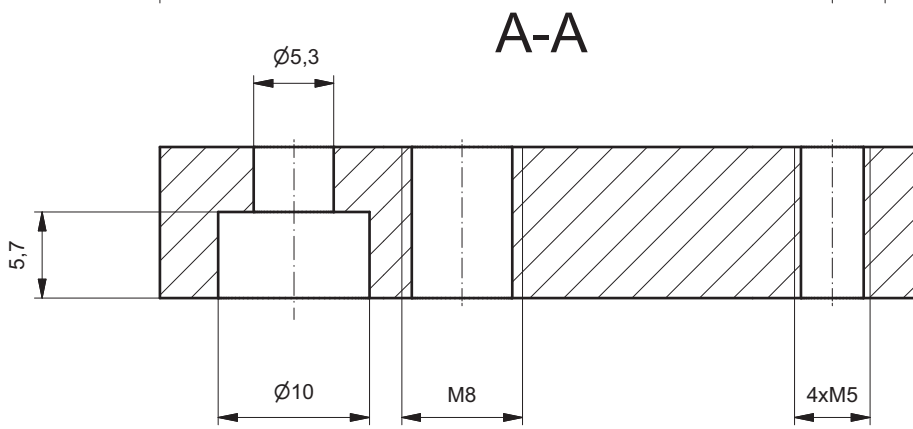
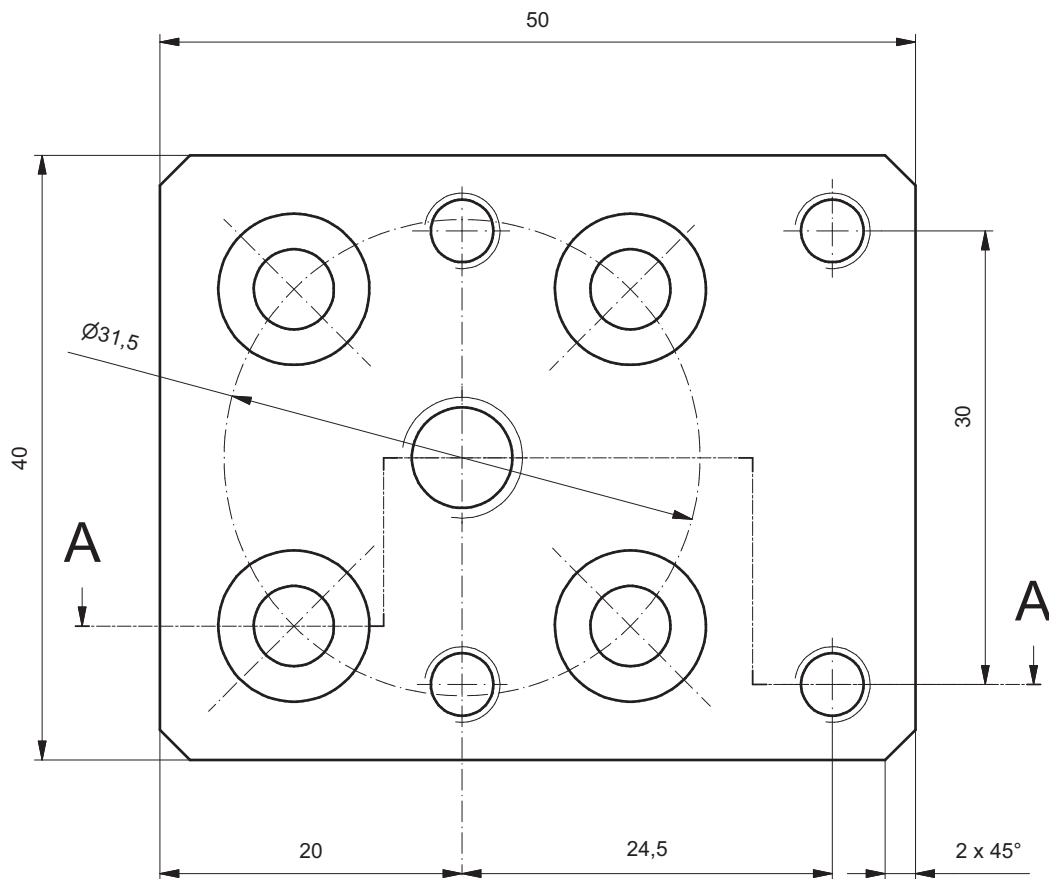
Fichier C:\Users\JB\Documents\Inventor\TD2014LeapMotion\VP003_Profile_L.idw
Datei

Hes·so VALAIS WALLIS

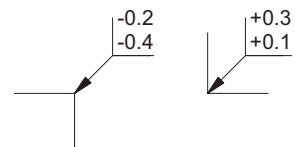
Annexe 10

P004

Ra 3,2



Tol. générales ISO 2768-mK



4	1	Aluminium	Support Pince
Pos.	Quantité	Matière	Dénomination / Caractéristiques
Pos.	Menge	Werkstoff	Benennung / Merkmale

TD: Application d'un Leap Motion à l'apprentissage de trajectoires	Dessiné Gezeichnet	JB Rebord	12.06.2014	Echelle Massstab 2:1
	Contrôlé Geprüft			

Fichier C:\Users\JB\Documents\Inventor\TD2014LeapMotion\P004_Support_Pince.idw
Datei

Tableaux

Série de mesures 1: Crayon immobile

	Longueur	Largeur	Position			
			Axe X	Axe Y	Axe Z	3 axes
Mesures [mm]	99.87011	5.75497	-109.64328	155.06680	-4.67852	XXX
	103.29660	6.82016	-109.83783	155.30551	-4.64401	XXX
	75.17144	6.86383	-109.69680	155.24825	-5.18119	XXX
	44.98291	6.47360	-109.65168	155.28539	-5.49239	XXX
	105.46230	6.80457	-109.49323	154.97484	-4.64426	XXX
Erreurs Absolues [mm]	14.11344	-0.78846	0.02128	-0.10936	0.24955	0.27329
	17.53993	0.27674	-0.17327	0.12935	0.28407	0.35700
	-10.58523	0.32041	-0.03224	0.07209	-0.25312	0.26515
	-40.77376	-0.06983	0.01288	0.10923	-0.56432	0.57494
	19.70562	0.26114	0.17134	-0.20132	0.28382	0.38786
Erreurs Relatives [%]	16.45754	-12.04965	-0.01941	-0.07047	-5.06385	0.14386
	20.45314	4.22923	0.15800	0.08336	-5.76426	0.18762
	-12.34333	4.89663	0.02940	0.04646	5.13620	0.13943
	-47.54588	-1.06716	-0.01175	0.07039	11.45107	0.30232
	22.97853	3.99095	-0.15624	-0.12974	-5.75916	0.20434

Tableaux

Série de mesures 2: Main géométrique immobile

	Longueur				
	Doigt 1	Doigt 2	Doigt 3	Doigt 4	Doigt 5
Mesures [mm]	43.46204	66.31373	71.78571	55.66061	44.86005
	43.55639	66.31377	71.67028	55.61494	44.92384
	43.56108	66.31606	71.66417	55.63205	44.85655
	43.55337	66.28516	71.67147	55.71989	44.82140
	43.42392	66.26803	71.68829	55.52008	44.86477
Référence [mm]	60.00000	70.00000	80.00000	70.00000	60.00000
Erreurs Absolues [mm]	-16.53796	-3.68627	-8.21429	-14.33939	-15.13995
	-16.44361	-3.68623	-8.32972	-14.38506	-15.07616
	-16.43892	-3.68394	-8.33583	-14.36795	-15.14345
	-16.44663	-3.71484	-8.32853	-14.28011	-15.17860
	-16.57608	-3.73197	-8.31171	-14.47992	-15.13523
Erreurs Relatives [%]	-27.56327	-5.26610	-10.26786	-20.48484	-25.23324
	-27.40602	-5.26604	-10.41215	-20.55009	-25.12693
	-27.39821	-5.26277	-10.41979	-20.52565	-25.23908
	-27.41105	-5.30691	-10.41066	-20.40016	-25.29767
	-27.62680	-5.33139	-10.38964	-20.68560	-25.22538

	Largeur				
	Doigt 1	Doigt 2	Doigt 3	Doigt 4	Doigt 5
Mesures [mm]	17.16299	12.07171	13.01091	15.79221	14.62052
	17.15040	12.05974	13.00466	15.82083	14.54697
	17.17179	12.06226	13.01083	15.82627	14.50654
	17.17563	12.07990	13.00006	15.80565	14.49210
	17.16213	12.09606	13.00448	15.81443	14.52665
Référence	18.00000	14.00000	14.00000	14.00000	13.00000
Erreurs Absolues [mm]	-0.83701	-1.92829	-0.98909	1.79221	1.62052
	-0.84960	-1.94027	-0.99534	1.82083	1.54697
	-0.82821	-1.93774	-0.98917	1.82627	1.50654
	-0.82437	-1.92010	-0.99994	1.80565	1.49210
	-0.83788	-1.90394	-0.99552	1.81443	1.52665
Erreurs Relatives [%]	-4.65007	-13.77350	-7.06496	12.80149	12.46557
	-4.72002	-13.85904	-7.10959	13.00596	11.89979
	-4.60117	-13.84101	-7.06548	13.04476	11.58875
	-4.57982	-13.71500	-7.14243	12.89747	11.47765
	-4.65486	-13.59959	-7.11083	12.96019	11.74346

	Angle			
	doigt1-doigt2	doigt2-doigt3	doigt3-doigt4	doigt4-doigt5
Mesures [°]	41.91080	29.24779	24.57124	29.22031
	41.92260	29.31902	24.45463	29.56803
	41.87931	29.32115	24.45942	29.40826
	41.96031	29.29418	24.50218	29.53241
	41.88287	29.29802	24.47770	29.69434
Référence [°]	60.00000	30.00000	30.00000	30.00000
Erreurs Absolues [°]	-18.08920	-0.75221	-5.42876	-0.77969
	-18.07740	-0.68098	-5.54537	-0.43197
	-18.12069	-0.67885	-5.54058	-0.59174
	-18.03969	-0.70582	-5.49782	-0.46759
	-18.11713	-0.70198	-5.52230	-0.30566
Erreurs Relatives [%]	-30.14867	-2.50737	-18.09588	-2.59895
	-30.12900	-2.26994	-18.48456	-1.43992
	-30.20116	-2.26284	-18.46862	-1.97245
	-30.06615	-2.35273	-18.32606	-1.55864
	-30.19521	-2.33995	-18.40766	-1.01886

Tableaux

	Position			
	Axe X	Axe Y	Axe Z	3 axes
Mesures [mm]	-1.78351	205.72034	-7.92628	XXX
	-1.76491	205.81677	-7.87696	XXX
	-1.69559	205.82881	-7.92046	XXX
	-1.78713	205.72163	-7.91187	XXX
	-1.86615	205.76512	-7.93702	XXX
Erreurs Absolues [mm]	-0.00405	-0.05019	-0.01177	0.05171
	0.01454	0.04624	0.03756	0.06132
	0.08387	0.05828	-0.00594	0.10230
	-0.00767	-0.04890	0.00265	0.04957
	-0.08669	-0.00541	-0.02250	0.08973
Erreurs Relatives [%]	0.22777	-0.02439	0.14865	0.02512
	-0.81732	0.02247	-0.47457	0.02977
	-4.71298	0.02832	0.07500	0.04966
	0.43091	-0.02377	-0.03343	0.02408
	4.87161	-0.00263	0.28435	0.04357

Tableaux

Série de mesures 3: Main géométrique en mouvement

	Longueur				
	Doigt 1	Doigt 2	Doigt 3	Doigt 4	Doigt 5
Mesures [mm]	46.18792	65.45124	65.97304	47.62197	41.61089
	46.29576	65.45505	65.90172	47.50467	41.47678
	46.18811	65.35944	65.88084	47.58891	41.69065
	46.00100	65.41335	65.96303	47.72346	41.57016
	45.90990	65.50327	66.25478	47.93328	41.37053
	45.99832	65.76401	66.75032	48.08489	41.31538
	45.74733	65.97769	67.34753	48.42397	41.67797
	45.37310	65.99289	67.76666	48.65180	42.48704
	45.48149	66.18102	68.01695	49.09868	42.76668
	45.44838	66.38626	68.27003	49.87273	42.91433
	45.25192	66.35471	68.41051	50.19178	43.78666
	44.95704	66.19694	68.50092	50.67943	44.64806
	44.59646	66.30656	68.83429	50.96733	44.62157
	44.46904	66.29556	69.07893	51.35444	44.99564
	44.55463	66.00983	69.09515	51.86905	46.01345
	44.55092	65.96069	69.39128	52.17541	46.72748
44.44199	66.01918	69.71977	52.64985	46.90926	
44.07771	65.97821	70.06838	53.09734	47.07445	
Référence [mm]	60.00000	70.00000	80.00000	70.00000	60.00000
Erreurs Absolues [mm]	-13.81208	-4.54876	-14.02696	-22.37803	-18.38911
	-13.70424	-4.54495	-14.09828	-22.49534	-18.52322
	-13.81189	-4.64056	-14.11916	-22.41110	-18.30935
	-13.99900	-4.58665	-14.03697	-22.27654	-18.42984
	-14.09010	-4.49674	-13.74522	-22.06672	-18.62948
	-14.00168	-4.23599	-13.24968	-21.91511	-18.68462
	-14.25267	-4.02231	-12.65247	-21.57603	-18.32203
	-14.62690	-4.00711	-12.23335	-21.34820	-17.51296
	-14.51851	-3.81898	-11.98305	-20.90132	-17.23332
	-14.55162	-3.61374	-11.72997	-20.12727	-17.08567
	-14.74809	-3.64529	-11.58949	-19.80822	-16.21334
	-15.04296	-3.80306	-11.49908	-19.32057	-15.35194
	-15.40354	-3.69344	-11.16571	-19.03267	-15.37843
	-15.53096	-3.70445	-10.92107	-18.64556	-15.00436
	-15.44537	-3.99017	-10.90485	-18.13096	-13.98655
	-15.44909	-4.03931	-10.60872	-17.82459	-13.27252
-15.55801	-3.98082	-10.28023	-17.35015	-13.09074	
-15.92230	-4.02179	-9.93162	-16.90266	-12.92555	
Erreurs Relatives [%]	-23.02013	-6.49823	-17.53370	-31.96861	-30.64851
	-22.84040	-6.49279	-17.62285	-32.13619	-30.87203
	-23.01982	-6.62938	-17.64895	-32.01585	-30.51559
	-23.33167	-6.55236	-17.54621	-31.82363	-30.71640
	-23.48351	-6.42391	-17.18153	-31.52389	-31.04913
	-23.33613	-6.05141	-16.56210	-31.30730	-31.14103
	-23.75446	-5.74616	-15.81559	-30.82290	-30.53672
	-24.37817	-5.72444	-15.29168	-30.49743	-29.18827
	-24.19752	-5.45569	-14.97881	-29.85903	-28.72220
	-24.25270	-5.16249	-14.66246	-28.75325	-28.47611
	-24.58014	-5.20756	-14.48686	-28.29746	-27.02223
	-25.07160	-5.43294	-14.37385	-27.60081	-25.58657
	-25.67256	-5.27634	-13.95714	-27.18953	-25.63072
	-25.88494	-5.29206	-13.65134	-26.63651	-25.00726
	-25.74228	-5.70024	-13.63107	-25.90136	-23.31091
	-25.74848	-5.77044	-13.26090	-25.46370	-22.12087
-25.93001	-5.68689	-12.85029	-24.78593	-21.81791	
-26.53716	-5.74541	-12.41453	-24.14666	-21.54259	

Tableaux

	Largeur				
	Doigt 1	Doigt 2	Doigt 3	Doigt 4	Doigt 5
Mesures [mm]	17.41902	12.89684	14.46844	15.01134	15.08057
	17.42155	12.89939	14.46595	15.01872	15.12860
	17.40002	12.90074	14.46954	15.00487	15.02921
	17.35771	12.89427	14.48814	15.01633	15.06860
	17.34550	12.88489	14.49482	15.18955	15.02176
	17.39387	12.88952	14.49108	15.38794	14.77282
	17.45837	12.89594	14.46853	15.50887	14.54269
	17.48143	12.90180	14.40147	15.47440	14.54206
	17.52538	12.89810	14.30339	15.42982	14.85358
	17.56416	12.88398	14.26675	15.30407	15.03362
	17.56577	12.87219	14.30334	15.14175	14.94864
	17.31632	12.85775	14.30448	15.14251	14.76344
	17.02527	12.84585	14.30566	15.14294	15.08463
	17.06905	12.84764	14.31715	15.18802	15.60238
	17.37988	12.86803	14.28946	15.15477	15.63533
	17.67108	12.87849	14.28841	15.06090	15.60401
	17.87273	12.88698	14.27130	15.18829	15.46736
	18.00967	12.89615	14.23906	15.46880	15.38704
Référence [mm]	18.00000	14.00000	14.00000	14.00000	13.00000
Erreurs Absolues [mm]	-0.58098	-1.10316	0.46844	1.01134	2.08057
	-0.57845	-1.10061	0.46595	1.01872	2.12860
	-0.59999	-1.09926	0.46954	1.00487	2.02921
	-0.64229	-1.10573	0.48814	1.01633	2.06860
	-0.65450	-1.11511	0.49482	1.18955	2.02176
	-0.60613	-1.11048	0.49108	1.38794	1.77282
	-0.54163	-1.10406	0.46853	1.50887	1.54269
	-0.51857	-1.09820	0.40147	1.47440	1.54206
	-0.47462	-1.10190	0.30339	1.42982	1.85358
	-0.43584	-1.11602	0.26675	1.30407	2.03362
	-0.43423	-1.12781	0.30334	1.14175	1.94864
	-0.68369	-1.14225	0.30448	1.14251	1.76344
	-0.97473	-1.15416	0.30566	1.14294	2.08463
	-0.93095	-1.15236	0.31715	1.18802	2.60238
	-0.62012	-1.13197	0.28946	1.15477	2.63533
	-0.32892	-1.12151	0.28841	1.06090	2.60401
	-0.12728	-1.11302	0.27130	1.18829	2.46736
	0.00967	-1.10385	0.23906	1.46880	2.38704
Erreurs Relatives [%]	-3.22766	-7.87972	3.34602	7.22385	16.00440
	-3.21363	-7.86148	3.32819	7.27656	16.37384
	-3.33325	-7.85183	3.35384	7.17761	15.60930
	-3.56829	-7.89805	3.48673	7.25952	15.91233
	-3.63609	-7.96506	3.53446	8.49681	15.55196
	-3.36741	-7.93203	3.50769	9.91384	13.63705
	-3.00906	-7.88613	3.34666	10.77766	11.86682
	-2.88093	-7.84427	2.86766	10.53145	11.86203
	-2.63676	-7.87069	2.16708	10.21302	14.25828
	-2.42136	-7.97159	1.90536	9.31478	15.64326
	-2.41239	-8.05580	2.16674	8.15536	14.98953
	-3.79825	-8.15894	2.17487	8.16077	13.56492
	-5.41517	-8.24396	2.18331	8.16389	16.03565
	-5.17194	-8.23115	2.26532	8.48585	20.01833
	-3.44511	-8.08548	2.06759	8.24833	20.27178
	-1.82732	-8.01079	2.06009	7.57786	20.03086
	-0.70708	-7.95014	1.93783	8.48777	18.97966
	0.05372	-7.88466	1.70756	10.49143	18.36183

Tableaux

	Angle			
	doigt1-doigt2	doigt2-doigt3	doigt3-doigt4	doigt4-doigt5
Mesures [°]	49.13789	27.13734	14.87173	52.40430
	49.07937	27.14661	14.88664	52.38014
	49.17401	27.20845	14.78554	52.59702
	49.06870	27.20781	14.95536	52.70806
	48.46227	27.41315	15.28488	52.08989
	47.82196	27.65625	15.72983	51.58668
	46.82119	27.94988	16.39951	51.05820
	46.17356	28.13479	16.98659	51.78715
	45.45912	28.02517	17.69492	51.42486
	44.96045	28.01205	18.09867	50.67748
	44.33623	28.16405	18.12305	50.02253
	44.33960	28.14656	18.34888	49.51377
	43.97892	28.15501	18.70618	49.51868
	43.50555	28.14560	19.18265	47.06823
	43.19736	28.21608	19.46914	25.84213
	42.92880	28.37803	19.84420	13.82862
	42.25647	28.49379	20.61341	8.25873
42.08961	28.63161	20.77955	8.83303	
Référence [°]	60.00000	30.00000	30.00000	30.00000
Erreurs Absolues [°]	-10.86211	-2.86266	-15.12827	22.40430
	-10.92063	-2.85339	-15.11336	22.38014
	-10.82599	-2.79155	-15.21446	22.59702
	-10.93130	-2.79219	-15.04464	22.70806
	-11.53773	-2.58685	-14.71512	22.08989
	-12.17804	-2.34375	-14.27017	21.58668
	-13.17881	-2.05012	-13.60049	21.05820
	-13.82644	-1.86521	-13.01341	21.78715
	-14.54088	-1.97483	-12.30508	21.42486
	-15.03955	-1.98795	-11.90133	20.67748
	-15.66377	-1.83595	-11.87695	20.02253
	-15.66040	-1.85344	-11.65112	19.51377
	-16.02108	-1.84499	-11.29382	19.51868
	-16.49445	-1.85440	-10.81735	17.06823
	-16.80264	-1.78392	-10.53086	-4.15787
	-17.07120	-1.62197	-10.15580	-16.17138
	-17.74353	-1.50621	-9.38659	-21.74127
-17.91039	-1.36839	-9.22045	-21.16697	
Erreurs Relatives [%]	-18.10352	-9.54221	-50.42758	74.68101
	-18.20105	-9.51130	-50.37786	74.60046
	-18.04332	-9.30518	-50.71486	75.32339
	-18.21883	-9.30729	-50.14878	75.69353
	-19.22954	-8.62282	-49.05040	73.63297
	-20.29673	-7.81248	-47.56725	71.95560
	-21.96468	-6.83372	-45.33496	70.19399
	-23.04407	-6.21735	-43.37804	72.62383
	-24.23480	-6.58277	-41.01692	71.41619
	-25.06591	-6.62651	-39.67112	68.92492
	-26.10628	-6.11985	-39.58982	66.74175
	-26.10067	-6.17812	-38.83705	65.04589
	-26.70181	-6.14995	-37.64606	65.06225
	-27.49076	-6.18133	-36.05784	56.89412
	-28.00441	-5.94640	-35.10286	-13.85956
	-28.45200	-5.40656	-33.85266	-53.90460
	-29.57255	-5.02071	-31.28863	-72.47090
-29.85065	-4.56130	-30.73485	-70.55657	

Tableaux

Série de mesures 4: Déplacement entre deux points connus

Point 1	Position [mm]			
	Axe X	Axe Y	Axe Z	3 axes
Mesures [mm]	-72.59761	168.83690	61.21156	XXX
	-72.98176	169.83887	62.66759	XXX
	-71.73979	169.25804	63.58793	XXX
Références [mm]	-72.00000	169.00000	65.00000	XXX
Erreurs Absolues [mm]	-0.59761	-0.16310	-3.78844	3.83875
	-0.98176	0.83887	-2.33241	2.66603
	0.26021	0.25804	-1.41207	1.45884
Erreurs Relatives [%]	0.83001	-0.09651	-5.82836	1.98171
	1.36356	0.49637	-3.58833	1.36587
	-0.36140	0.15269	-2.17241	0.74997

Point 2	Position [mm]			
	Axe X	Axe Y	Axe Z	3 axes
Mesures [mm]	54.27523	171.19983	-62.51819	XXX
	54.19479	168.72665	-52.63198	XXX
	54.62038	168.24312	-51.23231	XXX
Références [mm]	61.00000	169.00000	-71.00000	XXX
Erreurs Absolues [mm]	-6.72477	2.19983	8.48182	11.04550
	-6.80521	-0.27335	18.36802	19.59005
	-6.37962	-0.75688	19.76769	20.78543
Erreurs Relatives [%]	-11.02421	1.30167	-11.94622	5.80830
	-11.15609	-0.16175	-25.87045	10.59682
	-10.45840	-0.44786	-27.84182	11.28678

```
1 %Script calculant l'erreur des 4 séries de la première session
2
3 %Initialisation
4 clear all
5 clc
6
7 %Importation des fichiers xlsx
8 Tool = xlsread('Tool.xlsx');
9 MainImmobile = xlsread('Main Immobile.xlsx');
10 MainMouvement = xlsread('Main Mouvement.xlsx');
11 MainPositionsConnues1 = xlsread('Main Positions Connues 1.xlsx');
12 MainPositionsConnues2 = xlsread('Main Positions Connues 2.xlsx');
13
14 %*****
15 %Test avec outil (crayon)
16 %Dans ce test, le crayon est déplacé puis ramené à la même position entre
17 %chaque mesure
18 %Comparaison avec moyenne des mesures (erreur aléatoire)
19 %Longueur, épaisseur, position et angle
20
21 %Calcul des valeurs moyennes de chaque colonnes
22 ToolReferences = mean(Tool);
23
24 %Calcul des erreurs absolues et relatives par rapport aux références
25 for i=1:size(Tool,1)
26     for j=1:size(Tool,2)
27         ToolErreurAbs(i,j) = Tool(i,j)-ToolReferences(1,j);
28         ToolErreurRel(i,j) = (Tool(i,j)-ToolReferences(1,j))/ToolReferences(1,j) ←
*100;
29     end
30 end
31
32 %Erreur longueur
33 for i=1:size(Tool,1)
34     ToolErreurLongueurAbs(i,1) = ToolErreurAbs(i,2);
35     ToolErreurLongueurRel(i,1) = ToolErreurRel(i,2);
36 end
37
38 %Erreur largeur
39 for i=1:size(Tool,1)
40     ToolErreurLargeurAbs(i,1) = ToolErreurAbs(i,3);
41     ToolErreurLargeurRel(i,1) = ToolErreurRel(i,3);
42 end
43
44 %Erreur position (x,y,z)
45 for i=1:size(Tool,1)
46     %Pour chacune des dimensions (x,y et z)
47     for j=1:3
48         ToolErreurPositionAbs(i,j) = ToolErreurAbs(i,j+3);
49         ToolErreurPositionRel(i,j) = ToolErreurRel(i,j+3);
50     end
51     %Pour toutes les dimensions
52     j=4;
53     ToolErreurPositionAbs(i,j) = sqrt(ToolErreurAbs(i,4)^2+ToolErreurAbs(i,5) ←
^2+ToolErreurAbs(i,6)^2);
```



```
54     ToolErreurPositionRel(i,j) = ToolErreurPositionAbs(i,j)/sqrt(Tool(i,4)^2+Tool
(i,5)^2+Tool(i,6)^2)*100;
55 end
56
57 %Erreur absolue vecteur directeur (angle en degré)
58 for i=1:size(Tool,1)
59     ToolErreurAngle(i,1) = AngleEntre2Vecteurs(ToolReferences(1,7),ToolReferences
(1,8),ToolReferences(1,9),Tool(i,7),Tool(i,8),Tool(i,9));
60 end
61
62 %*****
63 %Test avec main géométrique
64 %Comparaison longueur et épaisseur des doigts et angle entre 2 doigts avec
65 %les valeurs connues de la main
66 %Comparaison position main avec moyenne des mesures
67
68 %Calcul des valeurs moyennes de chaque colonnes
69 MainImmobileReferences = mean(MainImmobile);
70 %Références connues (largeur et longueur des doigts)
71 %Pouce
72 MainImmobileReferences(1,15)=60;
73 MainImmobileReferences(1,16)=18;
74 %Index
75 MainImmobileReferences(1,24)=70;
76 MainImmobileReferences(1,25)=14;
77 %Majeur
78 MainImmobileReferences(1,33)=80;
79 MainImmobileReferences(1,34)=14;
80 %Annulaire
81 MainImmobileReferences(1,42)=70;
82 MainImmobileReferences(1,43)=14;
83 %Auriculaire
84 MainImmobileReferences(1,51)=60;
85 MainImmobileReferences(1,52)=13;
86
87 %Calcul des erreurs absolues et relatives par rapport aux références
88 for i=1:size(MainImmobile,1)
89     for j=1:size(MainImmobile,2)
90         MainImmobileErreurAbs(i,j) = MainImmobile(i,j)-MainImmobileReferences(1,j);
91         MainImmobileErreurRel(i,j) = (MainImmobile(i,j)-MainImmobileReferences(1,
j))/MainImmobileReferences(1,j)*100;
92     end
93 end
94
95 %Erreur position main (x,y,z)
96 for i=1:size(MainImmobile,1)
97     %Pour chacune des positions (x,y et z)
98     for j=1:3
99         MainImmobileErreurPositionAbs(i,j) = MainImmobileErreurAbs(i,j+1);
100        MainImmobileErreurPositionRel(i,j) = MainImmobileErreurRel(i,j+1);
101    end
102    %Pour toutes les dimensions
103    j=4;
104    MainImmobileErreurPositionAbs(i,j) = sqrt(MainImmobileErreurAbs(i,2)
^2+MainImmobileErreurAbs(i,3)^2+MainImmobileErreurAbs(i,4)^2);
```

```
105     MainImmobileErreurPositionRel(i,j) = MainImmobileErreurPositionAbs(i,j)/sqrt(
(MainImmobile(i,2)^2+MainImmobile(i,3)^2+MainImmobile(i,4)^2)*100;
106 end
107
108 %Erreur longueur doigts
109 for i=1:size(MainImmobile,1)
110     for j=1:5 %5 doigts...
111         MainImmobileErreurLongueurDoigtsAbs(i,j) = MainImmobileErreurAbs(i,15+(j-1)
*9);
112         MainImmobileErreurLongueurDoigtsRel(i,j) = MainImmobileErreurRel(i,15+(j-1)
*9);
113     end
114 end
115
116 %Erreur largeur doigts
117 for i=1:size(MainImmobile,1)
118     for j=1:5 %5 doigts...
119         MainImmobileErreurLargeurDoigtsAbs(i,j) = MainImmobileErreurAbs(i,16+(j-1)
*9);
120         MainImmobileErreurLargeurDoigtsRel(i,j) = MainImmobileErreurRel(i,16+(j-1)
*9);
121     end
122 end
123
124 %Erreur vecteur directeur entre 2 doigts (angle en degré)
125 %Ici on ne compare pas à la moyenne mais à la donnée connue
126 %(les angles entre 2 doigts sont des m multiples de 30 degrés)
127 for i=1:size(MainImmobile,1)
128     for j=1:4 %4 car il y a 4 espaces dans une main à 5 doigts
129         MainImmobileErreurAngleEntreDoigts(i,j) = AngleEntre2Vecteurs(MainImmobile
(i,20+(j-1)*9),MainImmobile(i,21+(j-1)*9),MainImmobile(i,22+(j-1)*9),MainImmobile(i,
20+j*9),MainImmobile(i,21+j*9),MainImmobile(i,22+j*9));
130         if (j==1)
131             AngleEntreDoigts=60;
132         else
133             AngleEntreDoigts=30;
134         end
135         MainImmobileErreurAngleEntreDoigtsAbs(i,j) =
MainImmobileErreurAngleEntreDoigts(i,j)-AngleEntreDoigts;
136         MainImmobileErreurAngleEntreDoigtsRel(i,j) =
(MainImmobileErreurAngleEntreDoigts(i,j)-AngleEntreDoigts)/AngleEntreDoigts*100;
137     end
138 end
139
140 %*****
141 %Test avec main géométrique en mouvement
142 %Comparaison longueur et épaisseur des doigts et angle entre 2 doigts avec
143 %les valeurs connues de la main
144 %Pas de comparaison de position: la main n'est jamais à la même position
145 %entre 2 frames
146
147 %Calcul des valeurs moyennes de chaque colonnes
148 MainMouvementReferences = mean(MainMouvement);
149 %Références connues (largeur et longueur des doigts)
150 %Pouce
```

```
151 MainMouvementReferences(1,15)=60;
152 MainMouvementReferences(1,16)=18;
153 %Index
154 MainMouvementReferences(1,24)=70;
155 MainMouvementReferences(1,25)=14;
156 %Majeur
157 MainMouvementReferences(1,33)=80;
158 MainMouvementReferences(1,34)=14;
159 %Annulaire
160 MainMouvementReferences(1,42)=70;
161 MainMouvementReferences(1,43)=14;
162 %Auriculaire
163 MainMouvementReferences(1,51)=60;
164 MainMouvementReferences(1,52)=13;
165
166 %Calcul des erreurs absolues et relatives par rapport aux références
167 for i=1:size(MainMouvement,1)
168     for j=1:size(MainMouvement,2)
169         MainMouvementErreurAbs(i,j) = MainMouvement(i,j)-MainMouvementReferences(1,
j);
170         MainMouvementErreurRel(i,j) = (MainMouvement(i,j)-MainMouvementReferences
(1,j))/MainMouvementReferences(1,j)*100;
171     end
172 end
173
174 %Erreur longueur doigts
175 for i=1:size(MainMouvement,1)
176     for j=1:5 %5 doigts...
177         MainMouvementErreurLongueurDoigtsAbs(i,j) = MainMouvementErreurAbs(i,15+(j-
1)*9);
178         MainMouvementErreurLongueurDoigtsRel(i,j) = MainMouvementErreurRel(i,15+(j-
1)*9);
179     end
180 end
181
182 %Erreur largeur doigts
183 for i=1:size(MainMouvement,1)
184     for j=1:5 %5 doigts...
185         MainMouvementErreurLargeurDoigtsAbs(i,j) = MainMouvementErreurAbs(i,16+(j-
1)*9);
186         MainMouvementErreurLargeurDoigtsRel(i,j) = MainMouvementErreurRel(i,16+(j-
1)*9);
187     end
188 end
189
190 %Erreur vecteur directeur entre 2 doigts (angle en degré)
191 %Ici on ne compare pas à la moyenne mais à la donnée connue
192 %(les angles entre 2 doigts sont des multiples de 30 degrés)
193 for i=1:size(MainMouvement,1)
194     for j=1:4 %4 car il y a 4 espaces dans une main à 5 doigts
195         MainMouvementErreurAngleEntreDoigts(i,j) = AngleEntre2Vecteurs
(MainMouvement(i,20+(j-1)*9),MainMouvement(i,21+(j-1)*9),MainMouvement(i,22+(j-1)*9),
MainMouvement(i,20+j*9),MainMouvement(i,21+j*9),MainMouvement(i,22+j*9));
196         if (j==1)
197             AngleEntreDoigts=60;
```

```
198     else
199         AngleEntreDoigts=30;
200     end
201     MainMouvementErreurAngleEntreDoigtsAbs(i,j) = ✓
MainMouvementErreurAngleEntreDoigts(i,j)-AngleEntreDoigts;
202     MainMouvementErreurAngleEntreDoigtsRel(i,j) = ✓
(MainMouvementErreurAngleEntreDoigts(i,j)-AngleEntreDoigts)/AngleEntreDoigts*100;
203     end
204 end
205
206 %*****
207 %Test avec positions absolues (référence connue) main géométriques
208 %Dans ce test, la main est placée à 2 positions dont la référence est connue
209 %Comparaison avec la référence et erreur de la différence des 2 positions
210
211 %Calcul des valeurs moyennes de chaque colonnes
212 MainPositionsConnuesReferences1 = mean(MainPositionsConnues1);
213 MainPositionsConnuesReferences2 = mean(MainPositionsConnues2);
214 %Références connues(position de la main)
215 %Position 1 Main
216 MainPositionsConnuesReferences1(1,2)=-72;
217 MainPositionsConnuesReferences1(1,3)=169;
218 MainPositionsConnuesReferences1(1,4)=65;
219
220 %Position 2 Main
221 MainPositionsConnuesReferences2(1,2)=61;
222 MainPositionsConnuesReferences2(1,3)=169;
223 MainPositionsConnuesReferences2(1,4)=-71;
224
225 %Calcul des erreurs absolues et relatives par rapport aux références
226 for i=1:size(MainPositionsConnues1,1)
227     for j=1:size(MainPositionsConnues1,2)
228         MainPositionsConnuesErreurAbs1(i,j) = MainPositionsConnues1(i,j)-✓
MainPositionsConnuesReferences1(1,j);
229         MainPositionsConnuesErreurRel1(i,j) = MainPositionsConnuesErreurAbs1(i,j) ✓
/MainPositionsConnuesReferences1(1,j)*100;
230         MainPositionsConnuesErreurAbs2(i,j) = MainPositionsConnues2(i,j)-✓
MainPositionsConnuesReferences2(1,j);
231         MainPositionsConnuesErreurRel2(i,j) = MainPositionsConnuesErreurAbs2(i,j) ✓
/MainPositionsConnuesReferences2(1,j)*100;
232     end
233 end
234
235 %Erreur position (x,y,z) par rapport à référence connue
236 for i=1:size(MainPositionsConnues1,1)
237     %Pour chacune des dimensions (x,y et z)
238     for j=1:3
239         MainPositionsConnuesErreurPositionAbs1(i,j) = ✓
MainPositionsConnuesErreurAbs1(i,j+1);
240         MainPositionsConnuesErreurPositionRel1(i,j) = ✓
MainPositionsConnuesErreurRel1(i,j+1);
241         MainPositionsConnuesErreurPositionAbs2(i,j) = ✓
MainPositionsConnuesErreurAbs2(i,j+1);
242         MainPositionsConnuesErreurPositionRel2(i,j) = ✓
MainPositionsConnuesErreurRel2(i,j+1);
```

```
243     end
244     %Pour toutes les dimensions
245     j=4;
246     MainPositionsConnuesErreurPositionAbs1(i,j) = sqrt
(MainPositionsConnuesErreurAbs1(i,2)^2+MainPositionsConnuesErreurAbs1(i,3)
^2+MainPositionsConnuesErreurAbs1(i,4)^2);
247     MainPositionsConnuesErreurPositionRel1(i,j) =
MainPositionsConnuesErreurPositionAbs1(i,j)/sqrt(MainPositionsConnues1(i,2)
^2+MainPositionsConnues1(i,3)^2+MainPositionsConnues1(i,4)^2)*100;
248     MainPositionsConnuesErreurPositionAbs2(i,j) = sqrt
(MainPositionsConnuesErreurAbs2(i,2)^2+MainPositionsConnuesErreurAbs2(i,3)
^2+MainPositionsConnuesErreurAbs2(i,4)^2);
249     MainPositionsConnuesErreurPositionRel2(i,j) =
MainPositionsConnuesErreurPositionAbs2(i,j)/sqrt(MainPositionsConnues2(i,2)
^2+MainPositionsConnues2(i,3)^2+MainPositionsConnues2(i,4)^2)*100;
250 end
251
252 %Erreur différence de position entre les 2 points
253 %Distance Théorique entre les deux points
254 for j=1:3
255     Distance2Points(j) = MainPositionsConnuesReferences2(1,j+1)-
MainPositionsConnuesReferences1(1,j+1);
256 end
257 DistanceTheoriqueEntreMains = sqrt(Distance2Points(1)^2+Distance2Points(2)
^2+Distance2Points(3)^2);
258
259 for i=1:size(MainPositionsConnues1,1)
260     for j=1:3 %X,Y et Z
261         MainPositionsConnuesErreurDifferencePosition(i,j) = MainPositionsConnues2
(i,j+1)-MainPositionsConnues1(i,j+1);
262         MainPositionsConnuesErreurDifferencePositionAbs(i,j) =
(MainPositionsConnues2(i,j+1)-MainPositionsConnues1(i,j+1))-Distance2Points(j);
263         MainPositionsConnuesErreurDifferencePositionRel(i,j) =
MainPositionsConnuesErreurDifferencePositionAbs(i,j)/Distance2Points(j)*100;
264     end
265     j=4;
266     MainPositionsConnuesErreurDifferencePosition(i,j) = sqrt((MainPositionsConnues2
(i,2)-MainPositionsConnues1(i,2))^2+(MainPositionsConnues2(i,3)-MainPositionsConnues1
(i,3))^2+(MainPositionsConnues2(i,4)-MainPositionsConnues1(i,4))^2);
267     MainPositionsConnuesErreurDifferencePositionAbs(i,j) = sqrt
((MainPositionsConnues2(i,2)-MainPositionsConnues1(i,2))^2+(MainPositionsConnues2(i,3)-
MainPositionsConnues1(i,3))^2+(MainPositionsConnues2(i,4)-MainPositionsConnues1(i,4))
^2)-DistanceTheoriqueEntreMains;
268     MainPositionsConnuesErreurDifferencePositionRel(i,j) =
MainPositionsConnuesErreurDifferencePositionAbs(i,j)/DistanceTheoriqueEntreMains*100;
269 end
270
271 %*****
272 %Suppression des variables inutiles
273 clear i;
274 clear j;
275 clear AngleEntreDoigts;
```

```
1 function errMoyTraj(fname,pos_sel,afficher)
2 %fonction qui cherche à corriger trajectoire en diminuant les écarts en
3 %tourant et déplaçant la forme de manière itérative
4
5
6 %***** Fonctionnement *****
7 %L'algorithmme doit avoir une trajectoire référence du parcourt fait par le
8 %bras robot et le fichier de mesure du parcourt
9
10 %Les points de mesures sont projetés sur la droite de référence la plus proche
11
12 %Correction de la rotation: on ramène tous les points au centre de masse du
13 %nuage de points et on évalue la rotation pour avoir le moins de décalage
14
15 %Correction de la translation déplacement du centre de masse pour avoir
16 %le décalage le plus petit
17
18 %On répète l'opération 2 fois (fonction itérative)
19
20
21 [path,traj_name,ext] = fileparts(fname);
22 traj_name = strcat(traj_name,ext);
23
24 X = dlmread(fname);
25 if nargin==1
26     pos_sel = 1:size(X,1);
27 end
28 X = X(pos_sel,:);
29 t = cumsum(X(:,1));
30 pts = X(:,2:4);
31 lines = dlmread('traj_ref_lines.txt');
32 ref = [lines(:,1:3); [lines(end,1) lines(end,2)+150 lines(end,3)]];
33 title_str = 'Erreur des points mesurés';
34
35 if strcmp(afficher,'ON')
36     figure;
37     plot3(ref(:,1),ref(:,2),ref(:,3),'-r',pts(:,1),pts(:,2),pts(:,3),'b.-');
38     legend('ref traj.','points mesurés');
39     title([fname ': ' title_str])
40     xlabel('x [mm]')
41     ylabel('y [mm]')
42     zlabel('z [mm]')
43 end
44
45 % trouve pour chaque point de mesure la référence la plus proche
46 % (projection)
47 [d,i,Pp] = P_min_ldist_and_proj(pts,lines);
48
49 show_error(t,pts-Pp,i,2,title_str,[0 0 1],fname,afficher);
50
51 T = [eye(3,3) [0 0 0]'; [0 0 0 1]];
52 Tpts = pts;
53 % méthode itérative (a chaque itération, on se rapproche de la référence
54 N_iter = 2;
55 for iter=1:N_iter
```

```

56     %Correction de la rotation
57     R = R_estim(Tpts,Pp);
58     % MAJ matrice transformation correction
59     T = [R [0 0 0]'; [0 0 0 1]]*T;
60     % calcul points transformés
61     Tpts = (T(1:3,1:3)*pts')'+ones(size(pts,1),1)*T(1:3,4)';
62     % calcul distances des points sur projection
63     [d,i,Pp] = P_min_ldist_and_proj(Tpts,lines);
64
65     % correction translation
66     tr = tr_estim(Tpts,Pp);
67     % MAJ matrice transformation correction
68     T = [eye(3,3) tr'; [0 0 0 1]]*T;
69     % calcul points transformés
70     Tpts = (T(1:3,1:3)*pts')'+ones(size(pts,1),1)*T(1:3,4)';
71     % calcul distances des points sur projection
72     [d,i,Pp] = P_min_ldist_and_proj(Tpts,lines);
73 end
74
75 %Affiche figure
76 title_str = 'Erreur des points corrigés';
77 if strcmp(afficher,'ON')
78     figure;
79     plot3(ref(:,1),ref(:,2),ref(:,3),'-r',[],[],[],Tpts(:,1),Tpts(:,2),Tpts(:,3),'-g' ↵
-g');
80     legend('trajet de référence.','points corrigés');
81     title([fname ': ' title_str])
82     xlabel('x [mm]')
83     ylabel('y [mm]')
84     zlabel('z [mm]')
85     disp(' ')
86     show_error(t,Tpts-Pp,i,4,title_str,[0 1 0],fname,afficher);
87     disp(' ')
88     disp('Matrice de transformation de correction:')
89     disp(T)
90 end
91
92 function R = R_estim(Ps,Pd)
93 ps = mean(Ps,1);
94 pd = mean(Pd,1);
95
96 Qs = Ps-ones(size(Ps,1),1)*ps;
97 Qd = Pd-ones(size(Pd,1),1)*pd;
98
99 H = Qs'*Qd;
100 [U,S,V] = svd(H);
101 X = V*U';
102 assert(abs(det(X)-1.0)<10*eps)
103 R=X;
104
105 function tr = tr_estim(Ps,Pd)
106 delta = Pd-Ps;
107 delta_norm = sqrt(sum(delta.^2,2));
108 delta_dir = delta./(delta_norm*ones(1,3));
109 tr = zeros(1,3);

```



```
110 for i=1:3
111     tr(i) = mean(delta(abs(delta_dir(:,i))>0.5,i));
112 end
113
114
115 function show_error(t,delta_pts,i,fig_h,title_str,d_color,fname,afficher)
116 d_norm_sqr = sum(delta_pts.^2,2);
117 d = sqrt(d_norm_sqr);
118 d_mean = mean(d);
119 d_rms = sqrt(mean(d_norm_sqr));
120 disp([title_str ':'])
121 fprintf(1,'Distance RMS: %f\n',d_rms);
122 fprintf(1,'Distance moyenne: %f\n',d_mean);
123 if strcmp(afficher,'ON')
124     figure;
125     subplot(2,1,1)
126     plot(t,delta_pts)
127     legend('x','y','z')
128     title_str=regexprep(title_str,'_','\\_');
129     title([fname ': ' title_str]);
130     ylabel('[mm]')
131     xlabel('temps [s]')
132     subplot(2,1,2)
133     plot(t,d,'Color',d_color)
134     ylabel('distance [mm]')
135     xlabel('temps [s]')
136 end
```

```
1 function tabErreur = ecartType(fname,pos_sel,axeRef,afficher)
2 %tab: nuage de points 3D
3 %axeRef=1:x
4 %axeRef=2:y
5 %axeRef=3:z
6 %refDebut: début de la droite sur l'axe de référence (avant la correction)
7 %refFin: fin de la droite sur l'axe de référence (avant la correction)
8
9
10 %***** Fonctionnement *****
11 %Importation depuis fichier
12
13 %On ramène l'axe de référence à x (en décalant les coordonnées)
14
15 %Régression du nuage de point à deux droite:
16 %droite1: y=f(x)
17 %droite2: z=f(x)
18
19 %on cherche l'angle de rotation par l'axe z puis on corrige la droite et
20 %le nuage de points
21 %idem avec angle de rotation par l'axe y
22
23 %Calcul des distances entre axeRef et points => tableau tabErreur
24
25 %Affichage de la droite et du nuage de points (original + correction)
26
27 %Calcul de l'écart type
28
29
30 %***** Fonctionnement *****
31 %Importation depuis le fichier
32 [path,traj_name,ext] = fileparts(fname);
33 traj_name = strcat(traj_name,ext);
34 X = dlmread(fname);
35 if nargin==1
36     pos_sel = 1:size(X,1);
37 end
38 X = X(pos_sel,:);
39 tab = X(:,2:4);
40
41 %Nombre de points
42 NbPoints=size(tab,1);
43
44 %Prise en compte de la référence
45 %tous les calculs suivants sont fait comme si x est l'axe de référence
46 switch axeRef
47     case 'Y'
48         %Si l'axe de référence est y: décalage des coordonnées des points
49         %Opération inverse après la régression
50         tab=[tab(:,2),tab(:,3),tab(:,1)];
51
52     case 'Z'
53         %Si l'axe de référence est z: décalage des coordonnées des points
54         %Opération inverse après la régression
55         tab=[tab(:,3),tab(:,1),tab(:,2)];
```

```
56 end
57 %Rajoute un zero à la fin de chaque ligne
58 tab=[tab ones(NbPoints,1)];
59
60 %Début et fin de la droite
61 refDebut=tab(1,1);
62 refFin=tab(NbPoints,1);
63
64 %Régression linéaire
65 %Définition des 2 pentes et des 2 ordonnées à l'origine (par rapport à
66 %l'axe de référence)
67 droite1 = polyfit(tab(:,1),tab(:,2),1);
68 droite2 = polyfit(tab(:,1),tab(:,3),1);
69
70 %2 points entre lesquels la droite est tracée
71 P1Droite=[refDebut, refDebut*droite1(1)+droite1(2), refDebut*droite2(1)+droite2(2), ✓
1];
72 P2Droite=[refFin, refFin*droite1(1)+droite1(2), refFin*droite2(1)+droite2(2), 1];
73
74 %Définition du premier angle (rotation axe z)
75 angle1=atan((P2Droite(2)-P1Droite(2))/(P2Droite(1)-P1Droite(1)));
76
77 %Correction 1
78 correction1=h_trans(P1Droite(1),P1Droite(2),P1Droite(3))*h_rotZ(-angle1)*h_trans(-✓
P1Droite(1),-P1Droite(2),-P1Droite(3));
79
80 %Correction 1 du tableau de points
81 for i=1:NbPoints
82     tabCorr(i,:)=(correction1*tab(i,:))';
83 end
84
85 %Correction 1 de la droite
86 P1DroiteCorr=(correction1*P1Droite)';
87 P2DroiteCorr=(correction1*P2Droite)';
88
89 P1DroiteCorr1=P1DroiteCorr;
90 P2DroiteCorr1=P2DroiteCorr;
91
92 %Définition du deuxième angle (rotation axe y)
93 angle2=atan((P2DroiteCorr(3)-P1DroiteCorr(3))/(P2DroiteCorr(1)-P1DroiteCorr(1)));
94
95 %Correction 2
96 correction2=h_trans(P1DroiteCorr(1),P1DroiteCorr(2),P1DroiteCorr(3))*h_rotY(angle2) ✓
*h_trans(-P1DroiteCorr(1),-P1DroiteCorr(2),-P1DroiteCorr(3));
97
98 %Correction 2 du tableau de points
99 for i=1:NbPoints
100     tabCorr(i,:)=(correction2*tabCorr(i,:))';
101 end
102
103 %Correction 2 de la droite
104 P1DroiteCorr=(correction2*P1DroiteCorr)';
105 P2DroiteCorr=(correction2*P2DroiteCorr)';
106
107 %Erreur
```

```

108 tabErreur=tabCorr-repmat([P1DroiteCorr],NbPoints,1);
109 %Mise à 0 de la première colonne
110 tabErreur(:,1)=zeros(NbPoints,1);
111 %Suppression de la 4ème colonne
112 tabErreur=tabErreur(1:NbPoints, 1:3);
113
114 %Décalage des coordonnées (inverse de l'opération au début de la fonction)
115 switch axeRef
116     case 'Y'
117         tab=[tab(:,3),tab(:,1),tab(:,2)];
118         tabCorr=[tabCorr(:,3),tabCorr(:,1),tabCorr(:,2)];
119         tabErreur=[tabErreur(:,3),tabErreur(:,1),tabErreur(:,2)];
120         P1Droite=[P1Droite(3),P1Droite(1),P1Droite(2)];
121         P2Droite=[P2Droite(3),P2Droite(1),P2Droite(2)];
122         P1DroiteCorr=[P1DroiteCorr(3),P1DroiteCorr(1),P1DroiteCorr(2)];
123         P2DroiteCorr=[P2DroiteCorr(3),P2DroiteCorr(1),P2DroiteCorr(2)];
124     case 'Z'
125         tab=[tab(:,2),tab(:,3),tab(:,1)];
126         tabCorr=[tabCorr(:,2),tabCorr(:,3),tabCorr(:,1)];
127         tabErreur=[tabErreur(:,2),tabErreur(:,3),tabErreur(:,1)];
128         P1Droite=[P1Droite(2),P1Droite(3),P1Droite(1)];
129         P2Droite=[P2Droite(2),P2Droite(3),P2Droite(1)];
130         P1DroiteCorr=[P1DroiteCorr(2),P1DroiteCorr(3),P1DroiteCorr(1)];
131         P2DroiteCorr=[P2DroiteCorr(2),P2DroiteCorr(3),P2DroiteCorr(1)];
132 end
133
134 %Afficher en 3D
135 %bleu: nuage de point original
136 if strcmp(afficher,'ON')
137     figure;
138     plot3(tab(:,1),tab(:,2),tab(:,3),'b.-');
139     hold on
140     %mauve: droite originale
141     plot3([P1Droite(1) P2Droite(1)],[P1Droite(2) P2Droite(2)],[P1Droite(3) P2Droite(3)],'m.-');
142     %rouge: droite modifiée
143     plot3([P1DroiteCorr(1) P2DroiteCorr(1)],[P1DroiteCorr(2) P2DroiteCorr(2)],[P1DroiteCorr(3) P2DroiteCorr(3)],'r.-');
144     %vert: nuage de point modifié
145     plot3(tabCorr(:,1),tabCorr(:,2),tabCorr(:,3),'g.-');
146     hold off
147     grid on
148     set(gca, 'DataAspectRatio', [1 1 1])
149     title([fname ': Trajet sur axe ' axeRef])
150     xlabel('X [mm]')
151     ylabel('Y [mm]')
152     zlabel('Z [mm]')
153     legend('points mesure','ajustement linéaire','correction droite // axe','points modifiés')
154 end
155
156 %Moyene (elle doit être égale à 0 pour chaque axe)
157 moyenneErreur=mean(tabErreur);
158
159 %Ecart Type

```

```
160 tabErreurCarr=tabErreur.^2;
161 sommeErrCarr=sum(tabErreurCarr);
162 ecartType=sqrt(sommeErrCarr/NbPoints);
163 disp(['Axe de référence ' axeRef ':'])
164
165 %Ecart maximal
166 ecartMax=max(abs(tabErreur));
167
168 switch axeRef
169     case 'X'
170         disp(['L'écart type en Y vaut: ' num2str(ecartType(2))]);
171         disp(['L'écart type en Z vaut: ' num2str(ecartType(3))]);
172         disp(['L'écart max en Y vaut: ' num2str(ecartMax(2))]);
173         disp(['L'écart max en Z vaut: ' num2str(ecartMax(3))]);
174     case 'Y'
175         disp(['L'écart type en X vaut: ' num2str(ecartType(1))]);
176         disp(['L'écart type en Z vaut: ' num2str(ecartType(3))]);
177         disp(['L'écart max en X vaut: ' num2str(ecartMax(1))]);
178         disp(['L'écart max en Z vaut: ' num2str(ecartMax(3))]);
179     case 'Z'
180         disp(['L'écart type en X vaut: ' num2str(ecartType(1))]);
181         disp(['L'écart type en Y vaut: ' num2str(ecartType(2))]);
182         disp(['L'écart max en X vaut: ' num2str(ecartMax(1))]);
183         disp(['L'écart max en Y vaut: ' num2str(ecartMax(2))]);
184 end
185
186 %Autre façon d'obtenir écart type (même réponse!)
187 %std(tabErreur,1)
188 %std = standard deviation = écart type
189 %parametre "1": prend en compte n points ("0": n-1)
```

Annexe 16

```
1 function tabNbElements = gauss(fname, tabErreur, axeCalcGauss, intervalle, limite,
affichage)
2
3 %Fonction qui afficher la courbe gaussienne d'un tableau d'erreurs et le
4 %nombre de points dans chaque intervalles (parametres de la fonction)
5
6
7 %***** Fonctionnement *****
8 %Recherche de l'axe de référence
9
10 %Calcul de l'écart type
11
12 %Tableau tabIntervalles entre -limite et limite avec incrément d'une
13 %valeur égale à intervalle
14
15 %Recherche du nombre de points de tabErreur dans chaque intervalle de
16 %tabIntervalles
17
18 %Courbe de Gauss
19
20 %Affichage
21
22
23
24 %Recherche de l'axe de référence
25 if (tabErreur(:,1)==zeros(size(tabErreur,1),1))
26     axeRef='X';
27 elseif (tabErreur(:,2)==zeros(size(tabErreur,1),1))
28     axeRef='Y';
29 elseif (tabErreur(:,3)==zeros(size(tabErreur,1),1))
30     axeRef='Z';
31 else
32     axeRef='axe de référence inconnu'
33 end
34
35 %Ecart Type selon axe demandé
36 NbPoints=size(tabErreur,1);
37 switch axeCalcGauss
38     case 'X'
39         axe=1;
40         tabErreurCarr=tabErreur(:,1).^2;
41     case 'Y'
42         axe=2;
43         tabErreurCarr=tabErreur(:,2).^2;
44     case 'Z'
45         axe=3;
46         tabErreurCarr=tabErreur(:,3).^2;
47 end
48 sommeErrCarr=sum(tabErreurCarr);
49 ecartType=sqrt(sommeErrCarr/NbPoints);
50
51 %Tableaux avec valeurs entre ~-limite et ~+limite avec 0 comme valeur
52 %centrale et in incrément de intervalle
53 tabIntervalles=[-intervalle*(round((limite/intervalle)+1)):intervalle:intervalle*
(round((limite/intervalle)+1))]';
```

```
54
55 %Recherches du nombre de valeur de tabErreur qui se rapprochent le plus
56 %des valeurs de tabIntervalles
57 for i=1:size(tabIntervalles,1)
58     compteur=0;
59     for j=1:size(tabErreur,1)
60         if ((tabErreur(j,axe)>=(tabIntervalles(i,1)-intervalle/2)) && (tabErreur(j,axe)<(tabIntervalles(i,1)+intervalle/2)))
61             compteur = compteur + 1;
62         end
63     end
64     tabNbElements(i,1)=compteur;
65 end
66
67 %Fonction de Gauss
68 %NbPointsDansLimites pas forcément égal à NbPoints si l'un d'entre eux est
69 %en dehors des limites
70 NbPointsDansLimites=sum(tabNbElements);
71 for i=1:size(tabIntervalles,1)
72     Gauss(i,1)=NbPointsDansLimites*exp(-((tabIntervalles(i,1)/ecartType)^2)/2)/ecartType/sqrt(2*pi);
73 end
74
75 %Affichage
76 %Rouge: courbe de Gauss
77 if strcmp(afficher,'ON')
78     figure;
79     plot(tabIntervalles, Gauss,'r.-')
80     hold on
81     %Bleu: nb de points par intervalle
82     plot(tabIntervalles, tabNbElements,'b.-')
83     hold off
84     grid on
85     title([fname ': Trajet sur axe ' axeRef ' - Erreur de l''axe ' axeCalcGauss])
86     xlabel('X [mm]')
87     ylabel('Y [Nombre de points]')
88     legend('Courbe de Gauss','Points par intervalle')
89 end
```