ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Retorno háptico para dispositivos táctiles mediante tecnología *Leap Motion*

AUTOR: **Rubén Laencina Escobar**

DIRECTOR: **Javier Garrigós Guerrero**

Septiembre / 2014

Proyecto Fin de Grado

RETORNO HÁPTICO PARA DISPOSITIVOS TÁCTILES MEDIANTE TECNOLOGÍA LEAP MOTION

RUBÉN LAENCINA ESCOBAR

Universidad Politécnica de Cartagena

Escuela Técnica Superior de Ingeniería de Telecomunicación
Septiembre, 2014

RUBÉN LAENCINA ESCOBAR

RETORNO HÁPTICO PARA DISPOSITIVOS TÁCTILES MEDIANTE TECNOLOGÍA LEAP MOTION

Dirigido por: **Prof. GARRIGÓS GUERRERO JAVIER**

Departamento de Electrónica, Tecnología de Computadoras y Proyectos

Escuela Técnica Superior de Ingeniería de Telecomunicación

Proyecto Fin de Grado

PROYECTO DESARROLLADO EN EL MARCO DEL
PROGRAMA ERASMUS

Degree Final Project

HAPTIC FEEDBACK FOR TACTILE DEVICES VIA LEAP MOTION TECHNOLOGY

Directed by: **Prof. LAURENT GRISONI**

Co-directed by: **Prof. FRÉDÉRIC GIRAUD**

Department of Informatics and Computer Science at Lille 1 University
France, July 2014

Agradecimientos

Hay varias personas cuyo apoyo durante el desarrollo de este proyecto final de grado me gustaría agradecer.

En primer lugar me gustaría agradecer a mi director de proyecto en la Universidad de Lille 1, Profesor Laurent Grisoni, su constante ayuda, disponibilidad, y sabio consejo, tanto en lo que al desarrollo del proyecto se refiere, así como en muchas otras ocasiones fuera del contexto de éste. Del mismo modo me gustaría agradecerle el haberme concedido la oportunidad de trabajar con el llevando a cabo este proyecto, en el seno y como parte activa, de su grupo de investigación en la Universidad de Lille 1, Francia. Asimismo me gustaría agradecerle al Profesor Asociado Fréderic Giroud su tiempo y ayuda siempre que requerí su consejo.

Agradezco igualmente a los profesores de la Universidad Politécnica de Cartagena, José María Molina García-Pardo y Javier Garrigós Martínez, respectivamente en calidad de Coordinador Erasmus y Director de Proyecto aquí en la Escuela Técnica Superior de Ingeniería de Telecomunicación de Cartagena, por todo su tiempo, esfuerzo y apoyo de principio a fin en esta aventura llamada Erasmus, con todo lo que ello supone.

Al margen de todo esto, y no podría ser de otra forma, agradezco especialmente y de manera muy fuerte a mi familia, por su infinita paciencia y todo su cuidado, no solo estos últimos 5 meses fuera de casa, sino a lo largo de toda mi vida. Su amor no tiene límites. A su manera, cada uno de ellos me ha hecho ser todo lo que soy a día de hoy. Ellos son mi todo.

Contenido

Resumen del TFG en Castellano

Contenido

Lista de Figuras

Capítulo 1
Introducción

Este proyecto se encuadra en las líneas de investigación del grupo MINT y del grupo STIMTAC, los cuales desarrollan su actividad en el seno de IRCICA, Telecom Lille, Universidad de Lille 1, Francia.

IRCICA (Institut de Recherche en Composants logiciel et matériel pour l'Information et la Communication Avancée) es un instituto de investigación interdisciplinario que involucra el CNRS (Centre National de la Recherche Scientifique) y la Universidad de Lille 1, Ciencias y Tecnología, el cual opera como un 'hotel de proyectos', combinando alrededor de 120 miembros de la facultad, investigadores, estudiantes, ingenieros y ténicos de 4 laboratorios asociados: IEMN (Institut d'Electronique de Microélectronique et de Nanotechnologie), LIFL (Laboratoire d'Informatique Fondamentale de Lille), Phlam (laboratoire de Physique des Lasers, Atomes et Molécules) y L2EP (Laboratoire d'Eelectrotechnique et d'Electronique de Puissance de Lille). La política científica de IRCICA tiene como objetivo poner en marcha proyectos de investigación interdisciplinarios que involucran diferentes comunidades en el instituto, en particular, hardware y software. Algunos de los proyectos actualmente en curso abarcan el estudio y desarrollo de nuevos dispositivos fotónicos, redes autónomas de ultra bajo consumo de energía, interfaces hápticas y arquitecturas bio-inspiradas de procesamiento de información.

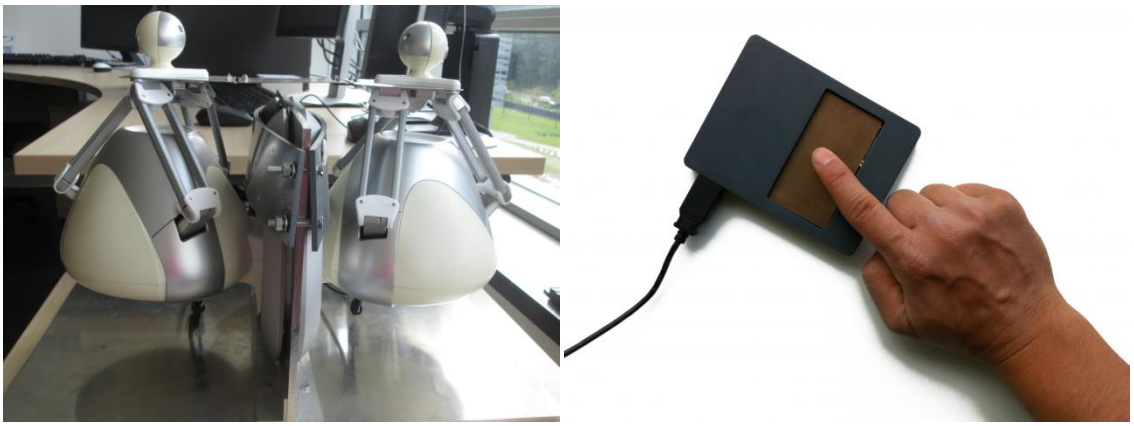
Por su parte, el MINT (Méthodes et outils pour l'Interaction à gestes) es uno de los grupos de investigación del LIFL, el cual presenta su núcleo principal de investigación en la interacción hombre-máquina (human-computer interaction, HCI), especialmente la interacción táctil y gestual. El *New Oxford American Dictionary* define gesto como *un movimiento de una parte del cuerpo, especialmente una mano o la cabeza, para expresar una idea o significado.* En el contexto particular de la HCI, el equipo MINT está interesado más específicamente en los movimientos que un sistema de computación puede detectar y a los cuales puede responder (un gesto de este modo puede ser visto como una función del tiempo en un conjunto de dimensiones detectadas que podrían incluir, pero no se limitan a la información posicional). De igual manera está interesado en el campo de la realidad virtual y en la interacción tanto en 2 como en 3 dimensiones (2D y 3D). También presenta una línea de trabajo sobre la interacción cerca de una pantalla, así como en el feedback táctil entre el usuario y ésta (tanto la parte de hardware a través de los especialistas en ingeniería eléctrica, como la parte que involucra hacer uso de la GUI).

En el campo de la realidad virtual, los dispositivos que proporcionan al usuario cualquier tipo de feedback, especialmente feedback táctil o en forma de fuerzas de diferente intensidad (force feedback), mejoran drásticamente la interacción hombre-máquina. Esto es obvio para personas con discapacidad visual, pero para personas 'sanas' es igualmente importante y pertinente experimentar un feedback háptico en según qué situaciones: en cirugía, en el aprendizaje de técnicas médicas de alta dificultad, en sistemas de drive-by-wire con el fin de hacer que la sensación mecánica experimentada sea lo más real posible, en juegos o en el diseño de nuevos productos en estrecha relación con la sensación de tacto, como

pueden ser ropa o superficies interiores de vehículos. Otro campo de gran interés para el feedback háptico cuando se trata de interacción hombre-máquina incluye teléfonos móviles, ordenadores portátiles o tabletas.

Por 'haptic feedback' (feedback háptico), se debe entender tanto force feedback como feedback táctil, o ambos. La sensación de tacto real es muy compleja e involucra simultaneamente la fuerza percibida por el usuario al entrar en contacto con un objeto cualquiera, como la percepción táctil que provocada por su superficie. Es por ello que actualmente existen gran cantidad de estudios que se centran en el acoplamiento entre el feedback kinestésico y cutáneo [1] [2].

En el marco del equipo de investigación MINT, y en el contexto de la tesis doctoral de Tao Zen, defendida en 2012, co-dirigida por Pr. Betty Semail (miembro del equipo MINT) y Zang Y. (Universidad de Beihang, Pekín) fue desarrollado un nuevo sistema basado en force feedback y feedback táctil con el fin de proporcionar al usuario la sensación de forma de superficies no planas, incluyendo la simulación de diferentes texturas [3] [4]. A fin de alcanzar este objetivo, se diseñó una plataforma cinemática 3 grados de libertad (3-DOF), construido a partir de dos dispositivos Novint Falcon, cuyos elemento final es un plato que es capaz de elevarse, girar y moverse en dos direcciones, x e y. Además, con el fin de cumplir el objetivo anterior, este último actuador es una placa de feedback táctil llamada STIMTAC (Stimulateur Táctil) [5] [6] desarrollada previamente de igual manera por el equipo MINT, en la que el usuario, al deslizar el dedo sobre ella, puede sentir diferentes texturas.



(a) Plataforma Novint Falcon 3-DOF (b) StimTac

Figura 1.1: Plataforma Novint Falcon 3-DOF y StimTac

Con el fin de mejorar y hacer lo más real posible la experiencia del usuario al interactuar con el dispositivo desarrollado aparece este proyecto. Dirigido por Laurent Grisoni, profesor de la Universidad de Lille 1 y principal responsable del equipo de investigación MINT, el objetivo principal de este proyecto es ampliar ese primer prototipo del dispositivo de interacción háptica diseñado por el equipo MINT, proporcionando una plataforma basada en tecnología Leap Motion de modo que la posición (dirección x e y), la elevación (dirección z) y la orientación (grados) del actuador final (Stimtac) se puedan predecir y adaptarse de forma continua a la posición del dedo del usuario con respecto a la forma virtual simulada que se supone ha de ser experimentada por el usuario.

Figura 1.2: Controlador Leap Motion

El presente informe muestra muy brevemente en las siguientes líneas: las principales tecnologías que intervienen a lo largo de todo el proyecto, el trabajo previo llevado a cabo por el equipo dentro de este campo, lo cual supone el punto de partida del presente proyecto, y el principal objetivo que se persigue con el mismo. De igual manera se muestra el sistema final diseñado, así como alguno de los experimentos llevados a cabo con tal fin. En última instancia, algunas conclusiones finales y una breve lista de futuras líneas de trabajo son sugeridas.

Véase **Memoria Original del TFG** para tener acceso al informe completo del presente proyecto.

Capítulo 2
Principales Tecnologías

Las principales tecnologías presentes en el desarrollo del presente proyecto, tanto en su vertiente hardware como software, son presentadas a lo largo de las siguientes líneas.

2.1 Leap Motion

Leap Motion, Inc. es una compañía americana con sede en San Francisco, California, EEUU, que fabrica y comercializa un dispositivo sensor de hardware informático conocido con por el mismo nombre que la empresa, Leap Motion, concebido para interactuar y controlar nuestro ordenador personal, de forma análoga a un ratón o un teclado, con la gran diferencia de que no requiere ningún tipo de contacto del usuario con éste.

El controlador Leap Motion ha sufrido gran cantidad de modificaciones desde que el primer prototipo fue diseñado en Agosto de 2011, hasta que el modelo final fue lanzado al mercado por primera vez en Julio de 2013.

Figura 2.1: Evolución del Leap Motion

El diseño final que ha llegado al usuario es esencialmente un pequeño periférico USB, no más grande que la palma de una mano, construido a partir de dos cámaras IR monocromáticas y 3 LEDs de infrarojo, el cual puede detectar tanto las dos manos como los diez dedos del usuario con una precisión de 0.01 mm y microsegundos, siempre que estas se encuentren dentro de su campo de visión, el cual lo compone el espacio tridimensional que se encuentra sobre él, siendo éste de aproximadamente un cubo de 2 pies de lado (60 cm).

Figura 2.2: Modo de empleo Leap Motion

El controlador Leap Motion viene acompañado igualmente por una serie de robustas plataformas llamada Leap Motion Airspace Home y Leap Motion Airspace Store donde gran cantidad de aplicaciones pueden ser encontradas para comenzar a familiarizarse con la tecnología Leap Motion. Asimismo presenta un completo SDK [SDKL] el cual contiene todas las librerías necesarias para fácilmente integrar la tecnología Leap Motion en cualquier proyecto.

En definitiva, este novedoso dispositivo, descrito por sus desarrolladores como *un dispositivo diminuto con gigantescas posibilidades*, es en realidad uno de los inventos actuales con más posibilidades de cambiar la manera en que entendamos y usemos la tecnología en los próximos años.

Ver Appendix A, Appendix C y Appendix D para una mayor información acerca de todo lo relacionado con el controlador Leap Motion y su SDK.

2.2 The Novint Falcon

Novint Technologies, Inc. es una sociedad constituida en Delaware y con sede en Albuquerque, New Mexico, EEUU, que diseña y desarrolla software y dispositivos hapticos y táctiles en 3D. En concreto en 2007 Novint desarrolló el Novint Falcon, el primer dispositivo háptico en 3D consumido a nivel mundial que permite al usuario hacer uso del sentido del tacto en informática y computación.

En esencia, el Novint Falcon (Figure 1.4.a) es un robot con conexión USB, que requiere de una fuente de alimentación externa, y que básicamente está formado por 3 brazos mecánicos y un agarre que permite al usuario controlarlos. Esos tres brazos dotan al robot de 3 grados de libertad, de manera que a medida que el usuario mueve el agarre en last res dimensiones (derecha-izquierda (eje-x) y delante-atrás (eje-y), como un ratón, pero también arriba-abajo (eje-z), funcionalidad que no presenta un raton convencional), el software que incorpora el Falcon le permite controlar en cada instante la posición del agarre, así como hacia dónde éste se mueve, creando y aplicando en consecuencia diferentes fuerzas que

el usuario puede sentir y experimentar cuando interactua con el dispositivo, y que son generadas por una serie de motores que éste presenta en su interior.

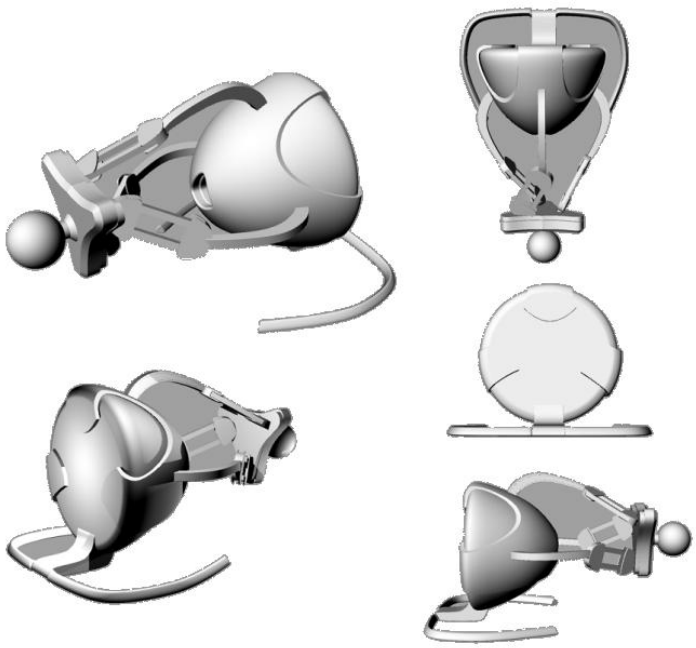


Figura 2.3: Novint Falcon 5-Vistas

Al igual que el Leap Motion, el Novint Falcon viene acompañado de un completo SDK y una serie de ejmplos que permiten una rápida familiarización con todas las posibilidades del dispositivo, así como su integración en cualquier proyecto.

El éxito del Novint Falcon de cara a usuarios, gamers e investigadores reside en dos claves fundamentales: su simplicidad y su bajo coste en comparación con otros dispositivos que se pueden encontrar en el mercado con similares características.

Véase **Memoria Original del TFG** para tener acceso a toda la información relativa al Capítulo 2 del presente informe.

Capítulo 3
Sistema Desarrollado

3.1 Introducción

Este proyecto se ha propuesto hacer uso de la novedosa tecnología Leap Motion como un sistema de control para el dispositivo de feedback háptico desarrollado por el grupo de investigación MINT (Sección 1.5), de manera que se consigue una adaptación continua de la posición, la elevacion y la orientación de una placa táctil (Stimtac), que hace el papel de actuador final del dispositivo, a la posición del dedo del usuario en cada instante, donde ningún tipo de contacto es considerado entre dicha placa táctil y el usuario. De acuerdo con esto (Sección 1.6) la función principal a ser controlada es la posición de ambos Novint Falcon, de los que se compone el dispositivo háptico, con el fin de que, conforme el dedo del usuario se acerca a la almohadilla tactil, ésta ya esté situada en la posición, elevación y orientación correcta, con respecto a la forma virtual que se supone está siendo simulada. Todo ello implica que el comportamiento de los Falcons Novint ha de ser totalmente preciso. Por otra parte, la integración entre estos y el controlador Leap Motion debe permitirnos tener una verdadera experiencia tactil al interactuar con el robot, además de ser todo desarrollado en tiempo real. Con este propósito un prototipo de "dispositivo háptico Leap Motion + Falcon Novint" ha sido desarrollado. A grandes rasgos éste consiste en una plataforma en la que el Leap Motion se enfrenta con el dispositivo de simulación háptico de manera que el área de interacción de ambos coinciden, ocupando así el mismo espacio. Además, el área de interacción del Leap Motion ha sido programada en un formato de imagen 2D, con una resolución dada en cada dirección (x e y), donde la información de cada píxel es la altura que la superficie a simular toma en esa posición. Cuando el dedo del usuario entra en la zona de interacción del Leap Motion, gracias a la forma en que ésta ha sido programada, es posible predecir el potencial punto de contacto del dedo del usuario con la almohadilla táctil en cada momento, es decir, el potencial píxel de contacto con la superficie virtual que está siendo simulada. Esto nos permite adaptar continuamente la Stimtac al dedo del usuario.

A tal fin varias funciones han sido desarrollado para inicializar y manejar ambos dispositivos, el Leap Motion y los Falcons Novint, así como para integrar ambos en la misma plataforma explicada anteriormente.

Se ha de resaltar nuevamente que dado que la plenitud del proyecto ha sido desarrollado sobre un primer prototipo del dispositivo háptico desarrollado hace tiempo por por laboratorio MINT, al tiempo que un nuevo prototipo más completo está siendo diseñado, el cual estará listo para el próximo año, como ha sido explicado en las secciones anteriores (Sección 1.6), todo el sistema desarrollado a lo largo de este proyecto se ha llevado a cabo de tal forma que se permita una adaptación lo más rápida, intuitiva y fácil de éste a esa nueva versión cuando esté lista.

3.2 Superficie Virtual Simulada

Como ya se ha dicho con anterioridad en varias ocasiones, el objetivo ultimo de este proyecto es ser capaces, a través de la interacción con el dispositivo háptico desarrollado, de similar superficies no planas en 3D de manera que el usuario es capaz de experimentarlas de manera táctil.

Estas superficies tridimensionales son definidas como una imagen en 2D (dimension en x e y) con una cierta resolución en cada dirección (la misma por defecto), donde la información guardada en cada pixel se corresponde con la altura que la superficie toma en ese punto, consiguiendo así las tres dimensiones que son simuladas por el dispositivo. Nuestra superficie en 3D se puede ver también como una imagen 2D en escala de grises, donde el nivel de gris es la altura que toma la superficie tridimiensional en cada punto de la imagen. Programablemente hablando esto es implementado mediante un array bidimensional donde la información de cada posición del array se corresponde con un pixel de la imagen en 2D.

La gran ventaja de establecer un format general para la definición de las superficies tridimensionales a similar es que cualquier imagen definida siguiendo dicho formato podría ser directamente simulada por el dispositivo sin la necesidad de realizar ningún otro tipo de cambio adicional sobre la misma.

Ver Apendix I para una mayor apreciación y entendimiento de cómo la superficie virtual a similar es definida.

3.3 Cálculo del Punto de Contacto

El controlador Leap Motion permite detectar las manos y dedos del usuario (la interacción del usuario con el dispositivo háptico se realize mediante un único dedo, normalmente el índice) y hacer uso de esta información con gran facilidad a partir de difetentes funciones incluidas en la API propia de Leap Motion. Sin embargo, la información del dedo del usuario por si solo es totalmente inútil. Esta información es de vital importancia, pero con respecto a la superficie virtual simulada, para lo cual el campo de interacción del Leap Motion es programado con dicha superficie.

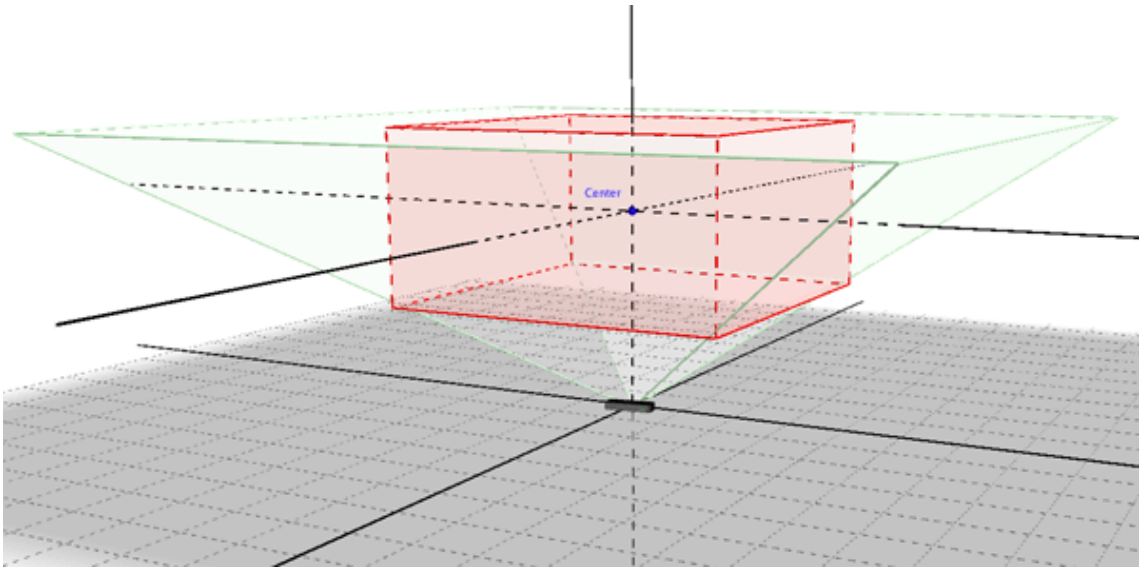


Figura 3.1: Definición del area de interacción del Leap Motion

A tal fin, y dado que como se ha dicho antes Leap Motion y Novint Falcon comparten area de interacción, el area de interacción del Leap Motion, la cual es mayor, es limitada al ROM (Rango de Movimiento) del dispositivo háptico, e igualmente programada con la supercie virtual definida anteriormente.

De esta manera, y tratando de la manera apropiada toda la información a proporcionada por el Leap Motion junto con la que ya poseemos es posible estimar el punto de contacto (pixel de contacto) del dedo del usuario con la superficie virtual.

Ver Appendix I para una mayor información de cómo es programada la superficie virtual.

3.4 Posición, Elevación y Orientación del Plato Táctil

De acuerdo con la sección anterior, efectivamente una vez que ha sido calculado el potencial punto de contacto del usuario con la superficial que está siendo simulada el siguiente paso es ser capaces de situar correctamente el plato tactil que forma parte de nuestro dispositivo háptico, con el cual finalmente se produce la interacción del usuario se produce, en el lugar correcto con antelación a que dicha interacción se produzca, de manera que la sensación táctil que experimente el usuario sea lo más real y fiel a la realidad posible.

En este sentido, para cada punto de contacto con la superficial simulada es necesario ser capaces de continuamente similar, con el plato táctil, el plano tangente dicho punto de manera que la sensación de tacto al interactuar con éste se lo más fiel posible [3] [4], siendo éste el principio fundamental a seguir (Figura 3.2) a la hora de calcular la posición, elevación y orientación del dispositivo táctil en cada instante.

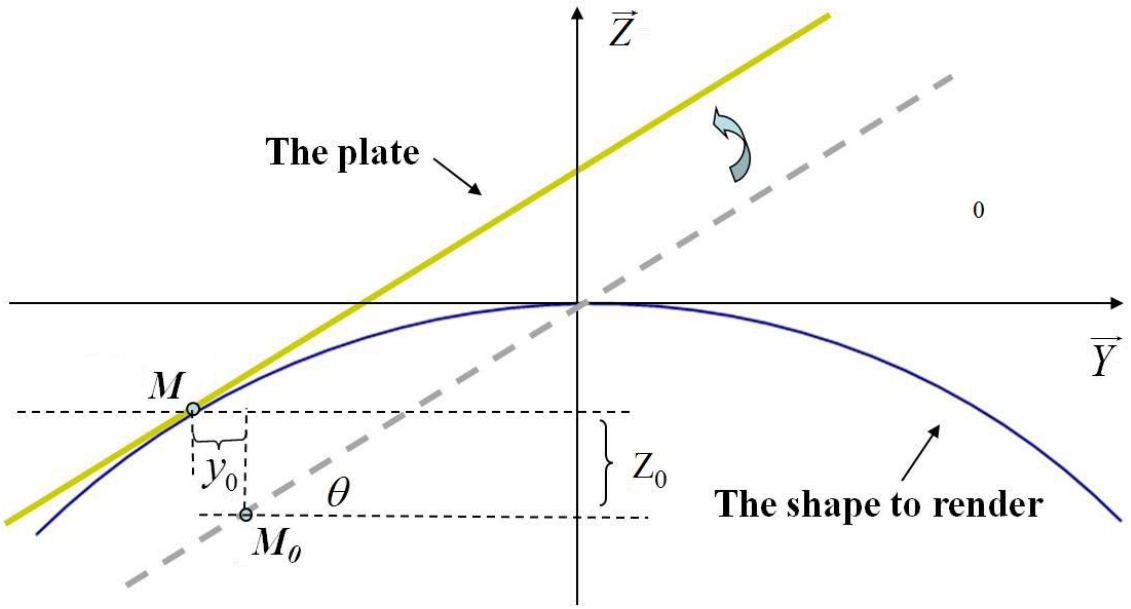


Figura 3.2: Principio Básico de Simulación de la Superficie Virtual

De manera indudable aquí se sitúa el punto crítico fundamental del éxito o el fracaso del proyecto. Ser capaces de establecer la correcta posición, elevación y orientación del dispositivo táctil en tiempo real, para cada punto de la superficie a

ser simulada, y siempre respetando dicho principio es el mayor de los retos propuesto por el presente proyecto.

Asimismo, dentro de lo que supone este problema, la estimación de la posición y la elevación de la placa táctil se puede obtener de manera relativamente sencilla a partir del pixel de contacto y la información almacenada en éste, sin embargo el cálculo de la orientación de la misma (grados, θ) es una tarea más compleja y la cual se ha de abarcar a partir del estudio de la posición relativa de los dos Novint Falcons de que se compone el dispositivo háptico con el que se está trabajando, tal y como se puede apreciar en la siguiente figura.

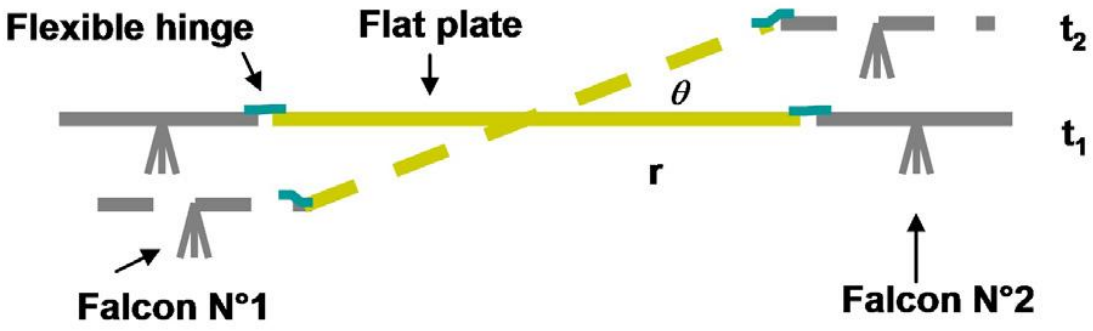


Figura 3.3: Control organization with the two "Falcon" systems

Finalmente y como es lógico, para un correcto funcionamiento del dispositivo y una experiencia háptica real por parte del usuario, las restricciones en los 3 grados de libertad del dispositivo han de ser respetadas por igual y de manera simultánea.

Ver Appendix I para tener acceso al código fuente completo.

3.5 Programa Prinicipal

La plenitud de la plataforma 'Leap Motion + Novint Falcon' desarrollada es manejada por una aplicación implementada en c++ con la versión 2005 de Visual Studio. En las siguientes líneas son presentadas tanto su organización general como sus partes más destacadas.

Inicialización

Una vez que el sistema se pone en marcha la primera cosa que se lleva a cabo es la creación de la superficie virtual a simuladar por el sistema. Inmediatamente después una inicialización de los Novint Falcon (Novint Falcon presenta sus propias funciones de inicialización suministradas por su SDK) es llevada a cabo. Por otro lado, ningún tipo de inicialización es necesaria para empleo del Leap Motion. Éste se encuentra listo para su utilización simplemente conectándolo al ordenador por su puerto USB.

Este tipo de inicialización permite un adecuado y fluido inicio del sistema con el fin de que el usuario puede empezar a utilizar los dispositivos sin ningún tipo de imprevisto.

Núcleo de la Aplicacion

El corazón de la aplicación implementada reside un un loop infinito que comienza cuando ésta es lanzada, y que únicamente termina (sin tener en cuenta eventos inesperados) cuando el usuario presiona la tecla 'enter' de su teclado o algunos de los botones presents en cualquiera de los dos Novint Falcon

En el seno de este loop tiene lugar la actividad de rastreo llevada a cabo por el Leap Motion así como cualquier otro tipo de actividad concerniente al acceso, tratamiento y computo tanto de la información arrojada por este como de la que se posee con anterioridad, tal y como ha sido explicado a lo largo de los apartados anteriores.

Sobre este loop de rastreo y tratamiento de información es implementado en segundo plano un segundo loop de alta velocidad para el control de la posición de ambos Novint Falcon, el cual centra su actividad en el cálculo de las fuerzas necesarias que se han de aplicar para el correcto posicionamiento del actuador táctil final en cada instante, de manera que el objetivo principal de este proyecto pueda ser llevado a cabo con éxito.

Véase Memoria Original del TFG para tener acceso a toda la información relativa al Capítulo 3 del presente informe.

Capítulo 4
Experimentos

4.1 Introducción

Gran cantidad de pruebas y experimentos han sido llevados a cabo antes de que la integración de los Novint Falcon y el Leap Motion fuese completa. Éstos son destacados en los siguientes apartados, y basicamente pueden ser divididos en dos grupos: experimentos iniciales y experimentos de sistema.

4.2 Experimentos Iniciales

Dentro de este grupo de experimentos se encuadran los experimentos llevados a cabo en las fases tempranas del proyecto con el Leap Motion, los Novint Falcon, y el dispositivo hàptico por separado, tanto para entrar en contacto y familiarizarnos con este tipo de tecnologías y dispositivos, así como para verificar la propia viabilidad del proyecto.

Algunas de las pruebas llevadas a cabo en esta fase son:

- Instalación de todo el software asociado tanto al Leap Motion como a los novint Falcon, incluyendo sus respectivos SDK.
- Ejecución y comprension de los diferentes tutoriales y ejemplos incluidos en los paquetes de instalación de los dispositivos.
- Testeo de las funcionalidades y prestaciones más simples de los dispositivos.

4.3 Experimentos de Sistema

En esta última fase de pruebas, una vez ya sobradamente conocidas las tecnologías involucradas en el proyecto, fueron llevados a cabo diferentes pruebas tanto para alcanzar el diseño del sistema propuesto a lo largo del proyecto, para determinar el comportamiento de éste una vez finalizado, así como para medir el grado de éxito total alcanzado a la hora de satisfacer los objetivos propuestos para el mismo.

Así, de todas los experimentos y medidas realizadas, las de mayor importancia para nosotros son aquellas encargadas de determinar el grado final de precisión alcanzado en la estimación de la posición de contacto del dedo del usuario con el plato táctil de que se compone la plataforma diseñada, respecto a los 3

grados de libertad del dispositivo háptico previamente diseñado (posición, elevación y orientación).

En esencia, estos experimentos consisten en la comparación de la posición teórica calculada en cada instante que debería ocupar el plato táctil en cada instante, a partir de la información proporcionada por el controlador Leap Motion, y la posición real en la que somos capaces de situar éste en tiempo real. Algunos de los resultados obtenidos son presentados en las siguientes gráficas:

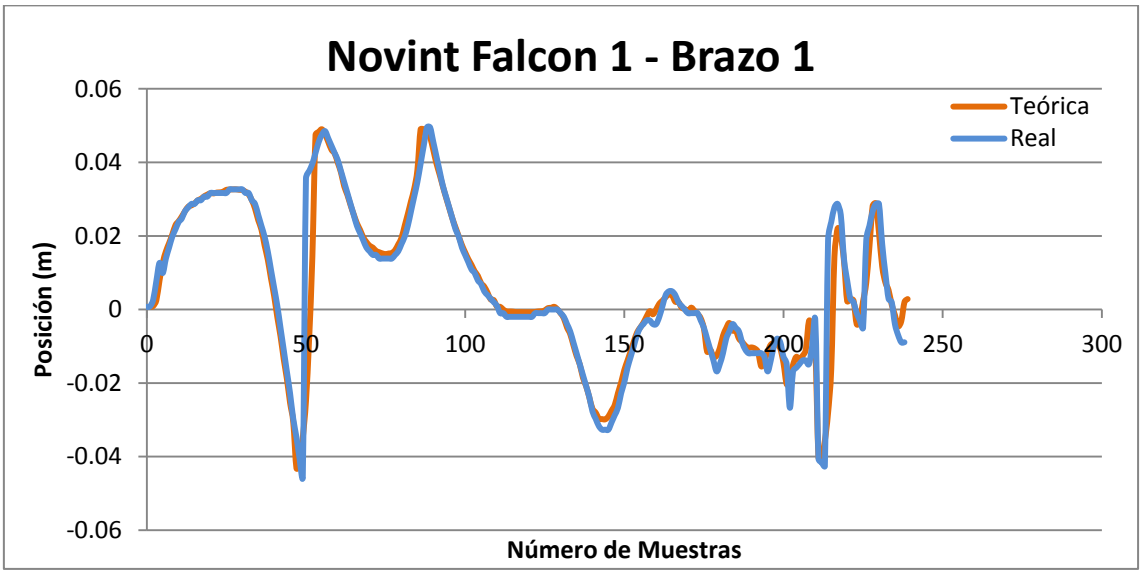


Figura 4.1: Medida Novint Falcon 1 – Brazo 1

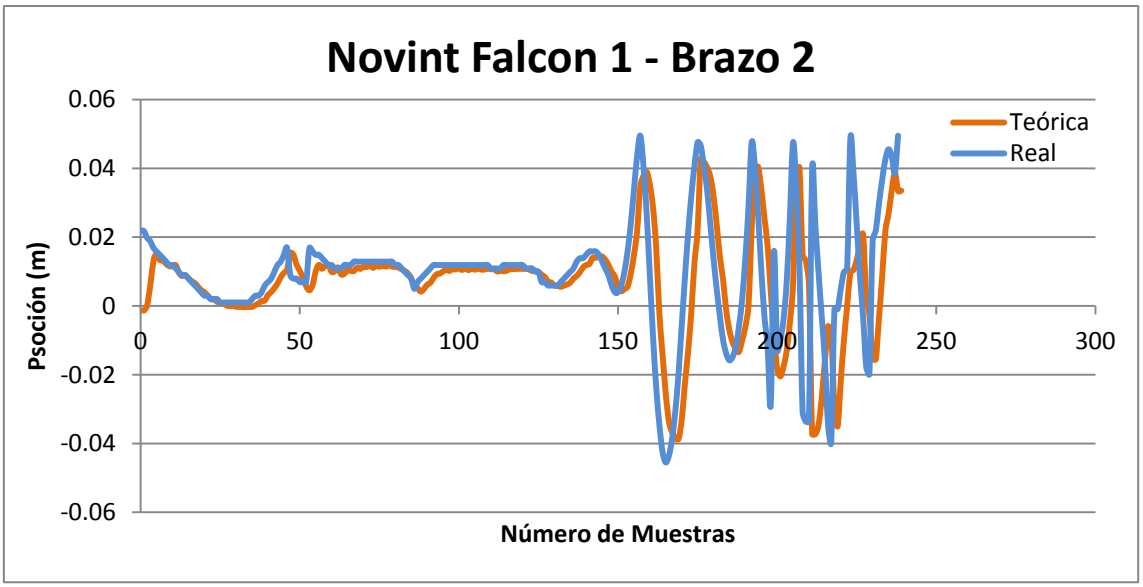


Figura 4.2: Medida Novint Falcon 1 – Brazo 2

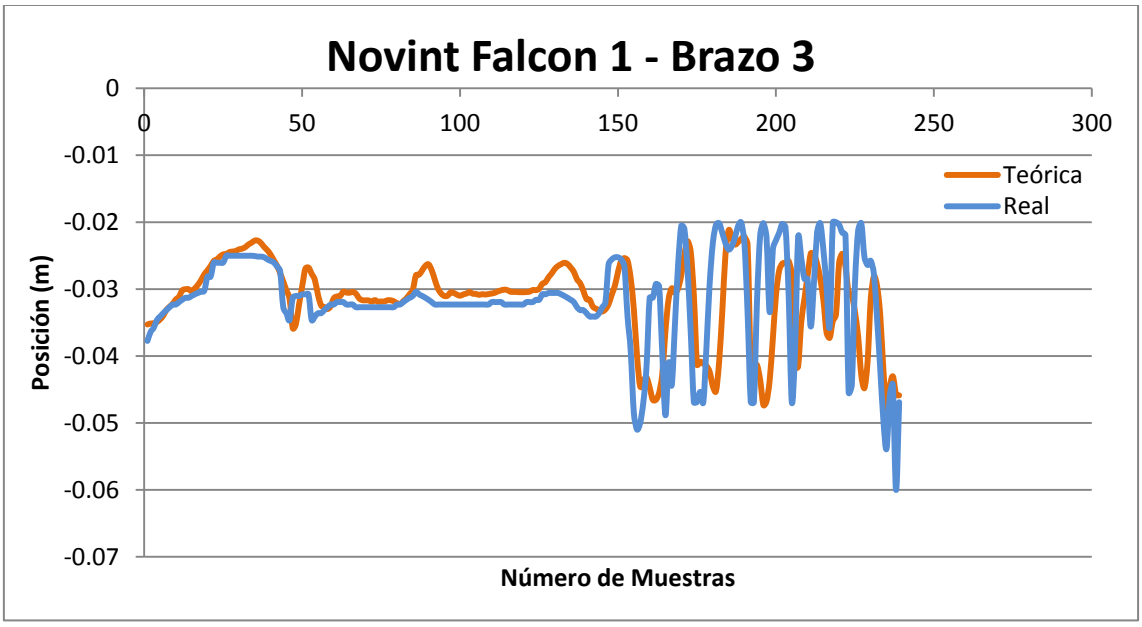


Figura 4.3: Medida Novint Falcon 1 – Brazo 3

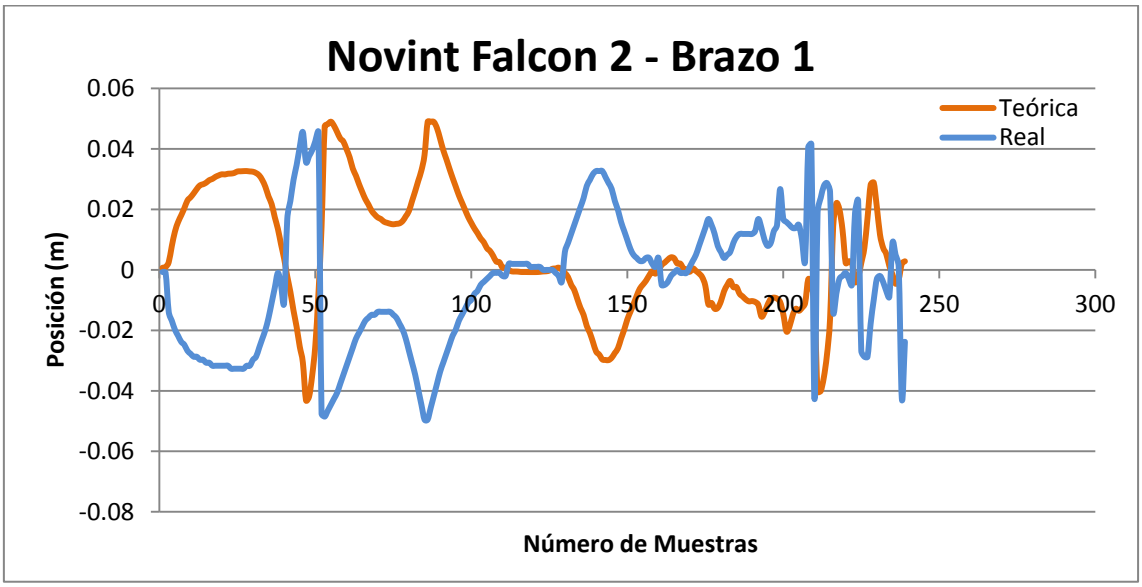


Figura 4.4: Medida Novint Falcon 2 – Brazo 1

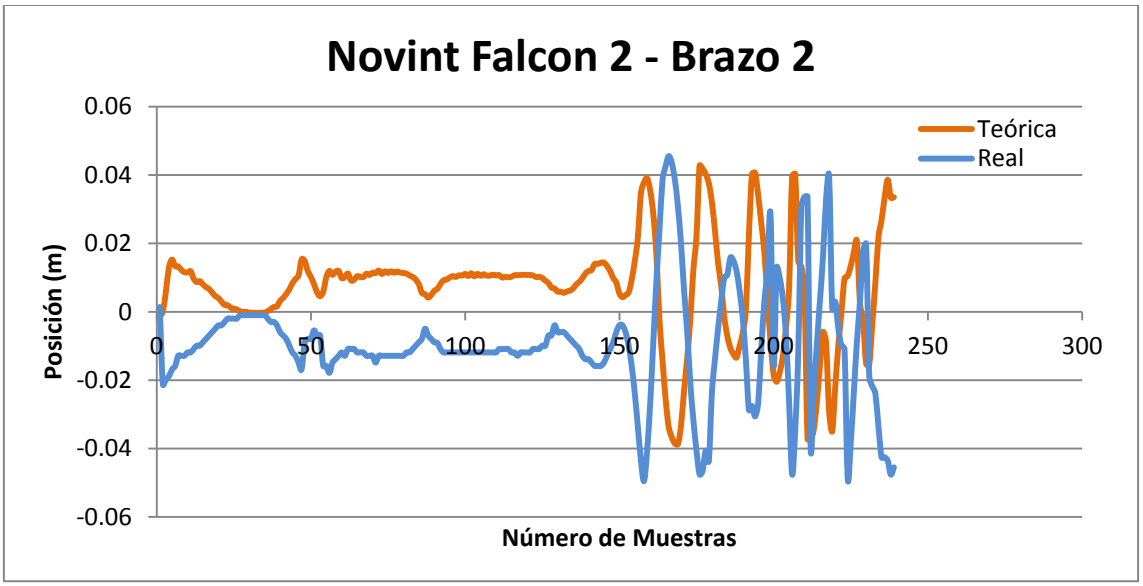


Figura 4.5: Medida Novint Falcon 2 – Brazo 2

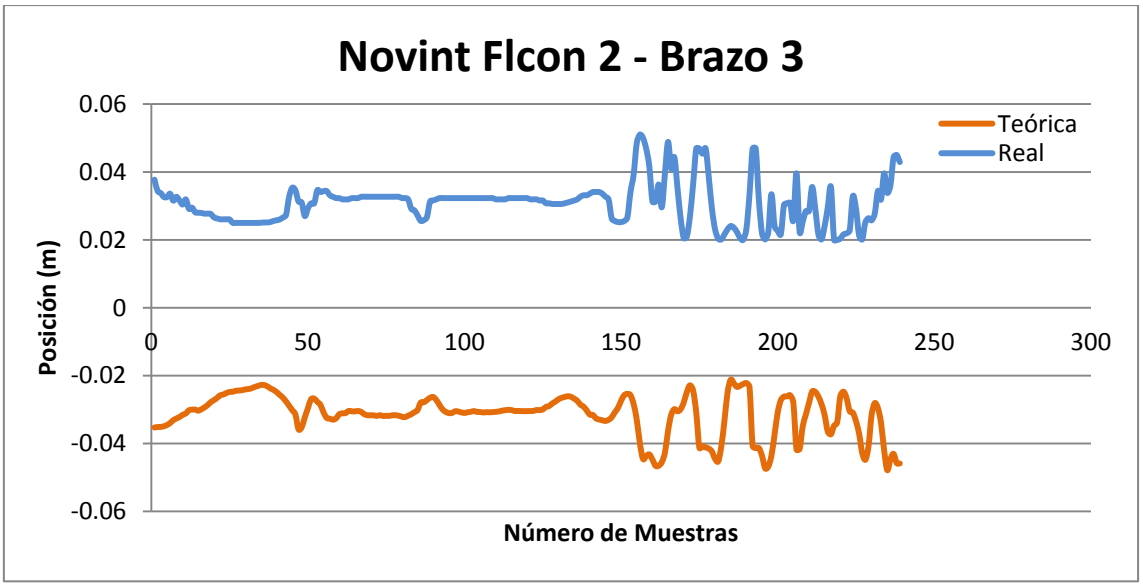


Figura 4.6: Medida Novint Falcon 2 – Brazo 3

En terminos reales, los resultados arrojan un valor del error medio obtenido a la hora de predecir y adaptar el plato táctil al dedo del usuario en tiempo real de 0.00438067 m para el Novint Falcon 1 y 0.00516276 m para el segundo. Es decir un error medio de en torno a 4-5 milímetros, lo cual puede considerarse como un comportamiendo más que aceptable, siendo este proyecto en definitiva la primera aproximación a integrar el Leap Motion con el dispositivo háptico diseñado por el grupo de investigación MINT de la Universidad de Lille 1.

4.4 Resultados

Todos los experimentos llevados a cabo han permitido testear exitosamente el software desarrollado así como la plataforma hardware diseñada para la

consecución del presente proyecto. De igual manera se ha de notar que en la realización de algunas de las pruebas finales para comprobar el desempeño de la misma, algunos comportamientos indeseados han sido observados, si bien eran esperados, de acuerdo al estudio inicial de viabilidad del proyecto que se realizó al comienzo del mismo y que se explican como resultado del uso del ambas tecnologías, tanto Leap Motion como Novint Falcon, con una finalidad totalmente distinta de para la que fueron pensadas.

Véase **Memoria Original del TFG** para tener acceso a toda la información relativa al Capítulo 4 del presente informe.

Capítulo 5
Conclusiones

5.1 Resumen

Este proyecto se ha dedicado a crear una integración entre el controlador Leap Motion y el dispositivo háptico diseñado a partir de dos Novint Falcon por el grupo de investigación MINT donde la tecnología Leap Motion permite añadir nuevas funcionalidades y mejorar las características generales y el comportamiento de este prototipo. Más precisamente, éste actúa como plataforma de control del dispositivo de interacción háptica de manera actuador final, que es una placa táctil desarrollada en el mismo laboratorio, se puede adaptar de manera continua a la interacción del usuario con respecto a una superficie virtual definida con anterioridad, para proporcionar al usuario una sensación táctil lo más real posible en términos tanto de forma como de textura.

De acuerdo con esto, haciendo uso del Leap Motion y del Novint Falcon SDK ha sido desarrollada con éxito una implementación que tiene como resultado una aplicación en tiempo real que satisface el objetivo principal del proyecto. Asimismo ha sido definido un formato general de entrada para las superficies virtuales que se deseen simular. A tal fin el área de interacción del Leap Motion ha sido programado para emular esa superficie virtual. A partir de éste una plataforma para predecir en cada instante el potencial punto de contacto del dedo del usuario con la superficie que se ha definido ha sido igualmente desarrollada, y toda esta información es tratado adecuadamente para que la placa táctil, que permite la simulación de textura, sea adaptada en tiempo real a la interacción del usuario, de modo que una sensación real de tacto es experimentada por el usuario al interactuar con el dispositivo.

Durante la fase de prueba del sistema, el código fuente desarrollado presenta resultados reales y fiables. De igual modo el comportamiento de la plataforma hardware cumple correctamente el objetivo principal del proyecto, si embargo algunos eventos inesperados han sido observados en el comportamiento final del sistema desarrollado. De acuerdo con ello, una breve lista de factibles soluciones ha sido proporcionada (Sección 4.5), las cuales, dado un prudencial margen de tiempo y una adecuada financiación, son totalmente realizables. Diversas sugerencias de posibles mejoras y líneas de trabajo futuro sobre el prototipo desarrollado son presentadas en la siguiente sección.

5.2 Trabajo Futuro

Como se ha resaltado en varias ocasiones a lo largo de este informe, la totalidad del trabajo desarrollado a lo largo de este proyecto ha sido realizado con un primer prototipo del dispositivo háptico desarrollado por el laboratorio MINT hace algunos años. Sin embargo, otro proyecto que tiene como objetivo la construcción de un nuevo prototipo, esta vez de 6-DOF, de un dispositivo similar al

aquí utilizado, con las mismas funcionalidades y extras, ya está teniendo lugar en el contexto de los grupos de investigación MINT y Stimtac. Es por ello que todas las sugerencias sobre posibles líneas de actuación futuras que son presentadas a continuación y con respecto a este proyecto, han sido realizadas en el marco de este nuevo prototipo de 6-DOF ya que, obviamente, una vez desarrollado, el viejo quedará obsoleto.

Además, teniendo en cuenta que este proyecto era sólo una "primera aproximación" a lo que significa el proyecto MINT global, debe tenerse en cuenta que todo el trabajo desarrollado en términos de procesamiento y tratamiento de datos, simulación virtual o integración Leap Motion - Novint Falcon está completamente sujeto a modificaciones y mejoras.

Software

En términos de Software algunos de los las posibles tareas que podrían ser llevadas a cabo en un futuro cercano son las siguientes:

- Desarrollo de una pequeña GUI, hacienda uso del software the Leap Motion que ya existe, de manera que el usuario pueda seguir en una pantalla, en tiempo real, su propia mano y en concreto su propio dedo con respect a la superficie virtual que esté siendo simulada por el dispositivo háptico, a fin de que la experiencia de usuario al interactuar con el dispositivo sea más intuitiva y sencilla.

- Consideración de la simulación de superficies virtuales de gran tamañp, es decir, superficies cuyo tamaño es mucho mayor que el ROM del dispositivo. Esto possible implementando de alguna manera una funcionalidad un zoom y unzoom con el software de Leap Motion (esto es algo ya utilizado en otras aplicaciones como Google Earth).

- Simulacion de superficies mixtas, es decir concavo-convexas, algo que todavía no ha sido realizado.

Hardware

En este campo, como ya hasido comentado, el desarrollo de un nuevo portotipo de 6 grados de libertad ya siendo llevado a cabo. En este context la necesidad principal reside en la integracion del controlador Leap Motion en esa nueva versión del dispositivo háptico, tarea que, de acuerdo con todo el trabajo presentado a lo largo del presente informe, debería ser fácilmente realizable.

Véase **Memoria Original del TFG** para tener acceso a toda la información relativa al Capítulo 5 del presente informe.

Memoria Original del TFG

Degree Final Project

HAPTIC FEEDBACK FOR TACTILE DEVICES VIA LEAP MOTION TECHNOLOGY

RUBÉN LAENCINA ESCOBAR
Telecommunication Engineering Degree

Department of Informatics and Computer Science at Lille 1 University
France, July 2014

RUBÉN LAENCINA ESCOBAR

HAPTIC FEEDBACK FOR TACTILE DEVICES VIA LEAP MOTION TECHNOLOGY

Directed by: **Prof. LAURENT GRISONI**

Co-directed by: **Prof. FRÉDÉRIC GIRAUD**

Acknowledgements

There are many people I would like to thank for their support during the development of this Degree Final Project. Firstly, I would like to thank my supervisor, Professor Laurent Grisoni for his support, assistance and wise feedback, within the project framework and out of it as well. I also would like to thank him for having given me the chance to work with him carrying out this project as part of his research team at Lille 1 University. I'd like to also thank Associate Professor Fréderic Giroud for his time and effort when I required his advice.

Aside this, and couldn't be otherwise, I specially and strongly thank my family, for their infinite patience and all the care taken of me along, not only these five month abroad, but my whole life. Their love has no limit. Each one of them in his own way has made me be all what nowdays I am. They are my everything.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 General Overview

This project was carried out within the framework of the INRIA MINT Project, and the STIMTAC Project of the IRCICA, Telecom Lille, University of Lille 1, France.

IRCICA (Institut de Recherche en Composants logiciel et matériel pour l'Information et la Communication Avancée) is an interdisciplinary research institute involving the CNRS (Centre National de la Recherche Scientifique) and the Université Lille 1, Science and Technology, which operates as a 'hotel of projects', combining about 120 faculty members, researchers, students, engineers and technicians from four partner laboratories: IEMN (Institut d'Electronique de Microélectronique et de Nanotechnologie), LIFL (Laboratoire d'Informatique Fondamentale de Lille), Phlam (laboratoire de Physique des Lasers, Atomes et Molécules) and L2EP (Laboratoire d'Eelectrotechnique et d'Electronique de Puissance de Lille). The scientific policy of IRCICA aims to initiate interdisciplinary research projects involving different communities in the institute, in particular hardware and software. Ongoing projects concern for example new photonic devices, autonomous networks ultra low energy, haptic touch interfaces and architectures bio-inspired information processing.

The MINT (Méthodes et outils pour l'Interaction à gestes) is one of the LIFL's research teams and it focuses on human-computer interaction (HCI), specifically the touch and gestural interaction. The New Oxford American Dictionary defines *gesture as a movement of part of the body, especially a hand or the head, to express an idea or meaning*. In the particular context of HCI, the MINT team is more specifically interested in movements that a computing system can sense and respond to (a gesture can thus be seen as a function of time into a set of sensed dimensions that might include but are not limited to positional information.) It is interested as well in the field of virtual reality and in the interaction of 2D and 3D content. It also has an axis about the interaction activity near a screen, as well as on the tactile feedback of effort (both the hardware part via those electrical engineering specialists, and the part to use the HCI).

In the field of virtual reality, force feedback and tactile feedback devices improve drastically the human machine interaction. It is obvious for visually impaired people but it is also pertinent to feel a haptic feedback for healthy people in other situations: for surgery, to learn difficult medical gestures, for drive-by-wire systems in order to render the real mechanical interaction feeling, for games, or to design new products in close relationship with touch feeling such as clothes or interior car surfaces. Another very important field of interest for haptic feedback is the Human-Machine interaction, including the cell phones, personal notebooks or tablets.

With 'haptic feedback', we may understand the force feedback or tactile feedback or both. The real sense of touch is very complex and involves simultaneously the force perception opposed by an object to a user, and the tactile

perception of its surface. Therefore, many studies now focus on the coupling between kinesthetic and cutaneous feedbacks [1] [2].

Within the framework of the MINT research team, and on the context of Tao Zeng PhD thesis, defended in 2012, coadvized by Pr. Betty Semail (MINT member) and Y. Zang (Beihang University, Beijing) a particular force and tactile feedback system was developed in order to provide the shape feeling of non flat surfaces to the user, including the surface texture rendering [3] [4]. Following this objective, they designed a kinematic 3 degrees of freedom platform (3-DOF), built from two Novint Falcon devices, whose end-effector is a plate which is able to rotate and to translate in *x* direction and *y* direction. Moreover, this end-effector is a tactile feedback plate called STIMTAC (Stimulateur Tactile) [5] [6] previously built in the MINT laboratory too, on which the finger may slide to feel different textures, so as to fulfill the previous objective.

In order to improve and to make as real as possible the user experience when interacting with the developed device appears this project. Directed by Laurent Grisoni, professor of the University of Lille 1 and main responsible of the MINT research team, the main goal of this project is to extend that early prototype of haptic interaction device designed by the MINT team by providing a Leap Motion control platform so that the position (x-direction and y-direction), the elevation (z-direction) and the orientation (degrees) of the end-effector pad (Stimac) can be continuously predicted and adapted to the position of the user finger regarding to the simulated virtual shape that is supposed to be rendered.

The current report accuracy presents in the following lines the main technologies that are involved along the whole project, the previous work carried out by the team within this field, which represents our project start point, and the main goal to be achieved. It will also be shown in detail the designed final system as well as all the experiments carried out in order to succeed. Some final conclusions and a short list of suggested future work is provided as well.

1.2 Leap Motion

Leap Motion, Inc. is an American company (Figure 1.1) with headquarters in San Francisco, California, United States, that manufactures and markets a computer hardware sensor device conceived to support hand and finger motions as input to a computer, analogous to a mouse, but requiring no hand contact or touching.

Figure 1.1: Leap Motion Company Logo

Founded in 2010 by Michael Buckwald and David Holz, and operating in quiet since then, after several millionaire investments from different well-known American investors Leap Motion publicly announced its first product, originally called *The Leap*, on May 21, 2012. The device started full-scale shipping in July 2013, at a price of 89.99€.

Considered by many people as the ground breaking release of this century turning any desktop, and even laptop, into technology with the next level of user interface, motion control, Leap Motion Inc's technology and its device, the Leap Motion Controller (Figure 1.2) hardly bigger than the palm of your hand, attaches to any Apple or PC screen with the Leap Motion software running, just plug it into the USB and without special adapters, senses your hands and fingers following their every move. It tracks both hands and all 10 fingers with pinpoint precision and incredible speed and lets them move in all that wide-open space between you and your computer. So you can do everything without in fact touching anything. This novel device, described by its developers as *a tiny device with huge possibilities*, is actually up to change the way we understand and use technology in the next few years.

Figure 1.2: The Leap Motion Controller

Thought in the very first moment to replace the mouse or the keyboard when it comes to interact and control our personal computer, the true is that it takes some time to get used to it and in the end some of its applications are not as intuitive as they were supposed to be, plus the fact that the necessity of having the hands in the air to control it can be very tiring in long periods of interaction.

If there is something all experts in this field agree is the fact that the maximum level of performance and the most surprising and amazing application of the Leap won't be as a substitute of the keyboard, mouse, stylus, or trackpad as we know them, but working with them and with other different software and hardware as well. The possibilities offered by this device regarding to *Play, Create and Explore* are currently deeply untold.

1.3 Haptic Feedback

Haptics (from Greek ἅπτω = 'I fasten onto, I touch') is related to any form of nonverbal communication involving touch, that is, involving the touch feeling. Nowadays, within the framework of virtual reality, the human-machine interaction which relies mainly on the hearing and sight may be enhanced by adding the touch sense.

Haptic devices or force feedback devices, add a third sense (sight, sound and *touch*) to what a user experiences when controlling an electronic device. This "touch" feedback can assist an operator by giving touch feedback when an object is grasped by a gripper for example, or further immerse a gamer into a computer generated world by providing touch feedback when their player is injured. This touch feedback can come in the form of vibrations or pushing / pulling actions supplied by a built in motor in the controller. The first force feedback systems were built in the 1950's to handle radioactive waste through teleoperation, while the first gaming application was developed by SEGA in 1975 for a motorcycle driving game. "Moto-Cross" implemented haptic feedback by vibrating the handle bars if a vehicle collision occurred. Car driving games later incorporated haptic feedback into the steering wheel to simulate a collision or hard turn which introduced a type of "push" haptic feedback.

(a) Delta 6 DOF (b) Omega 3 DOF (c) Omni 6 DOF

Figure 1.3: Leading Haptic Devices

In 1997, the "Rumble Pak" was used with the "Nintendo 64" and allowed haptic feedback controllers to be used in the average home. These smaller controllers were purely hand held, therefore limited to a "vibration" type of haptic feedback, generally when your player took damage or bumped into an object. In addition to a strong gaming interest, research institutions used haptic feedback devices in teleoperation applications such as material handling and medical surgery. Unfortunately, these devices cost between $10 000 and $25 000 US! Three of the most popular devices used in research are shown in Figure 1.3.

1.4 The Novint Falcon

Novint Technologies, Inc. is a corporation incorporated in Delaware and based in Albuquerque, New Mexico, United States which designs and builds haptic, or 3D touch, devices and software. In 2007 Novint developed the Novint Falcon, the world's first consumer 3D touch device, which allows users to use their sense of touch in computing. Novint has two primary areas of focus, video games and professional uses of its technology. In video games, the Novint Falcon can be used to feel objects and events in the game, giving the player a more immersive experience. In the professional applications group in Novint, called the Advanced Products Group (APG), Novint's technology has been used to add the sense of touch to a variety of professional applications and projects.

The Novint Falcon (Figure 1.4.a), Novint's flagship consumer product, is a 3 degrees of freedom (3-DOF) force feedback haptic interface, released by Novint at the Consumer Electronics Show (CES), it was primarily conceived for the gaming industry, more precisely to replace the mouse in video games and other applications (its name comes from the fact that the falcon is a predator of the mouse), making it the cheapest and most versatile haptic feedback device offered on the consumer market. Novint has also developed several grip accessories. On the consumer side, Novint developed a pistol grip (Figure 1.4.b), which is the shape of a pistol handle and attaches to the Falcon in place of the spherical grip. It has a main trigger button, and 3 side buttons and it was intended for use in First Person Shooter (FPS) games, allowing gamers to immerse themselves even further into the gaming experience, but is generally an ergonomic grip that can be used for many applications.

Due to its low cost and simplicity, since it was released the Novint Falcon has been well received by gamers, enthusiasts and researchers alike.

(a) Novint Falcon with Standard Grip (b) Optional Pistol Grip

Figure 1.4: Novint Falcon and Pistol Grip

1.5 Previous Work

Leap Motion

This is the first project at the MINT Lab to deal with Leap Motion technology, therefore much of the information in this report has been presented in a manner which allows future projects a head start in their development. All information to install, setup and operate the Leap Motion Controller has been included in a thorough guide. Information on the internet is sparse at best but developers can ask for advice at Leap Motion forums.

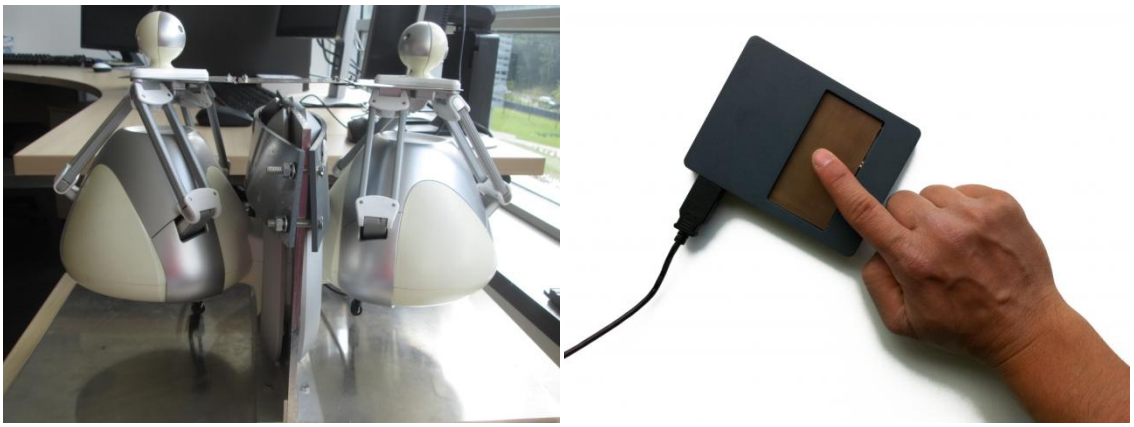
Novint Falcon

As it's been previously seen, in the field of virtual reality the improvement of the user immersion requires the use of human-machine interfaces capable of transmitting but also rendering the realistic information.

With this idea in mind, around 2009 within the framework of L2EP and MINT laboratory took up a PhD thesis which aim was to develop a haptic device which can reproduce the kinesthetic and tactile feedbacks, which is the reproduction of virtual touch sensations. Precisely, that work aimed to simultaneously simulate the shape and the texture sensation. For this purpose, they designed a kinesthetic platform to simulate the shape and a tactile plate to simulate the texture, and then integrated them in a compact device. For the simulation of shape, they proposed to render the curvature of shape by exploring a continuous surface that can be orientated, be elevated and be translated while always remaining tangent to a virtual shape at the contact point, in order to respect the principal source of information in the perception of shape (the orientation of the local surface). For the simulation of texture, a tactile plate previously developed in L2EP was integrated, which provides variable and controlled friction.

Finally, they integrated the platform and the tactile plate in a device coupling kinesthetic and tactile feedbacks that allows us to simultaneously reproduce the sensations of shape and fine texture. Several evaluations were also performed in the context of Tao Zeng PhD to know and test the performance of the device [3] [4].

The kinesthetic platform developed to simulate the shape is a kinematic 3 degrees of freedom platform designed using two Novint Falcon (Figure 1.5.a) whose end-effector is a plate which is able to rotate and to translate in *x* direction and *y* direction thanks to the simultaneous control of both Novint Falcon. Moreover, this end-effector is an ultrasonic tactile feedback plate called Stimtac (Figure 1.5.b) previously built in the same laboratory, on which the finger may slide to feel different textures. This tactile pad, actuated by piezoelectric ceramics and basing on a squeeze film effect between a plate and a finger is capable of giving programmable tactile sensations by modifying the friction coefficient on the surface of the plate, that is, between the plate and the user finger, thanks to controlled ultrasonic vibrations applied to the piezoelectric actuators which the haptic device is built from [5] [6].



(a) 3-DOF Novint Falcon Platform (b) StimTac

Figure 1.5: 3-DOF Novint Falcon Platform and StimTac

1.6 Goal

This project aims to extend the early prototype of haptic interaction device designed by the MINT team (Figure 1.5.a) by providing a Leap Motion control platform so that the position (x-direction and y-direction), the elevation (z-direction) and the orientation (degrees) of the end-effector pad (Stimac) can be continuously predicted and adapted to the position of the user finger regarding to the simulated virtual shape which is supposed to be rendered, leading in this way the reality of the user experience when interacting with the developed device to a next level.

Although preliminary results regarding to the success of the device when it comes to simulate concave and convex surfaces have been obtained on the context of Tao Zeng PhD this prototype is fully far away from a commercial version which could be placed in the market.

One of the most obvious shortcomings that can be felt when testing this first prototype is that it doesn't support the user with any kind of feedback until the user finger is in contact with the end-effector tactil pad. The device remains motionless while the user is not touching the StimTac. Plus, when the touch exists there is a little delay from the moment the contact takes place to the moment the device gives a response; delay corresponding to the computational time the implemented software and hardware need to calculate the right feedback. As a consequence the touching sense is poor and not in real time. In addition, because of the physical

constriction of the prototype only 2D shapes can be rendered, which is not enough since the final aim of the global MINT project is to develop a device able to simulate virtual surface in 3D.

According to this, more specifically the purpose of this project involves the achievement of the following tasks:

- Extend the feedback from 2D to 3D shapes.
- Make so that finger feedback does not feel any discontinuity (the less possible) when touching the control surface.
- Continuous and precise feedback in real time.

Even thought having said that because of the physical constriction of the prototype only 2D shapes can be simulated the purpose of this project is to create an integration between the Novint Falcon haptic device and the Leap Motion Controller that can provide 3D shapes rendering. This is going to be in this way because parallel to the development of this project within the framework of the MINT lab it's being carried out another project that, following the same principles and in a period of 1 year, aims to develop a new prototype of this shape rendering device but this time with 6 degrees of freedom, so that all the physical constrictions of the old prototype won't be a problem anymore.

Thus, and this is a point of maximum importance when carrying out this project, all the system developed along this project, although will be developed over the old prototype, it will be done in a way that could be totally adaptable and integrated to the new hardware prototype by doing the least possible modifications and in the easiest way.

It also must be noticed, as usual when it comes to human-machine interaction, that is mandatory for the final developed system to support the user with a feedback in real time and with the least possible discontinuities so that the sense of touch and the haptic experience of the user when interacting with the device is as natural, precise, fluid and real as possible. A low performance level in any of these fields wouldn't be accepted in any case.

1.7 Workflow

The general workflow of this project it is very easy and intuitive. It will begin with the approaching of the user's finger to the StimTac. Once the user's finger goes into the Leap Motion's workspace and always without touching the tactil plate, computing the finger's position regarding to the virtual shape to render the right position of the Novint Falcons are calculated so that the Stimtac 's position, elevation and orientation is properly set with anticipation to the touch of the user. It remains like this while the contact between the user finger and the tactil pad doesn't take place and the user finger is within the Leap Motion's workspace (if the user finger is out of it, the 3-DOF device is programmed to not expect an interaction with the user, therefore in the last position it has when the user finger was fithin the device's workspace). Once the user touch the pad and the contact exists, the control of this one and in the end of the Novint Falcons is handled by the StimTac and the contact point in its surface (this stuff is not handled by this project). Once the StimTac-user interaction stops, the control of the device it is again done by the Leap Motion in the way previously explained, and so on until the system execution is totally stopped. This is possible in two ways: pressing the

'enter' button of the keybord of the host computer of the application, or pressing any of the buttons of any of both haptic devices.

All communications, control schemes and haptic functions will be carried out in the main program which will rely on the Novint Falcon SDK and the Leap Motion SDK.

1.8 Practical Applications

The potential benefits and applications of using such a novel technology and hardware as Leap Motion and its controller, in continuous improvement and development, are very numerous and varied.

Beyond its tiny size, the 3D provided workspace and the fact that the Leap Motion Controller includes fast refresh rate, sub millimeter accuracy and real-time control possibilities, its advantages in a role of controller for the haptic device developed by MINT lab, with tracking and predicting functionalities, before other available options that could have been developed the same behavior such as heat sensors are that Leap Motion provides a robust and already tested hardware which means a significant saving of time when considering the whole context of the project, because of the non necessity of developing a new one. Integrating heat sensors in the developed prototype would have implicated an important hardware work by the team which would have meant a non negligible delay for the project, which was limited to a duration of 5 months.

Moreover, Leap Motion provides a fully completed software platform which potential usefulness opens many possibilities with regard to possible future work (Section 5.2) within the framework of this project and this is deeply interesting since we are dealing with a medium-long term project.

Aside the better user experience achieved along this project by using Leap Motion technology, it brings a wide field of new improvements for the haptic feedback device taken into account along the whole project that with other implementations such us the previously commented based on heat sensors would not have been even considered. The development of an accurate GUI so that the user experience becomes more intuitive and easy, or considering the problem of big-size virtual surfaces simulation are just two examples of all these new possibilities.

The only pursued aim with all this is the achievement and development of the most accurate and complete haptic surface and texture rendering device so that in the future a real marketable version could be developed.

1.9 Report Structure

Chapter 1: Introduction

A brief background of how Haptics have developed is covered along with an introduction of the Novint Falcon and Leap Motion technolgy, and the General Workflow. A small section on the practical applications is discussed.

Chapter 2: Equipment

Any hardware or software used in this project is described in this chapter.

Chapter 3: Developed System

The finished system developed along this project is described in full here.

Chapter 4: Experiments

Any experiments carried out are discussed which includes Leap Motion controller, Novint Falcon SDK, the 3 DOF Haptic device and the final system developed.

Chapter 5: Conclusion

A summary of the tasks completed and suggestions for future work can be found in this last chapter.

Appendices

Any setup and installation instructions have been included in the appropriate Appendix so that this project can be quickly re-created and built upon in the future. All the source code developed along this project has been included in this section for reference.

Chapter 2
Equipment

Any hardware or software used in this project is fully described in the following lines:

2.1 Hardware

Leap Motion

The Leap Motion Controller has suffered a lot of modifications since the first prototype was designed in August 2011 until a final model was placed in the market for the first time by July 2013 (Figure 2.1).

Figure 2.1: Evolution of the Leap Motion Controller

The final consumer design is a small USB peripheral device built using two monochromatic IR cameras and three infrared LEDs. The LEDs generate a 3D pattern of dots of IR light and the cameras generate almost 300 frames per second of reflected data, which is then sent through a USB cable to the host computer, where it is analyzed by the Leap Motion controller software using "complex math" in a way that has not been disclosed by the company, in some way synthesizing 3D position data by comparing the 2D frames generated by the two cameras.

Figure 2.2: Hardware Leap Motion Controller

According to its creators the Leap Motion Controller is 200 times more sensitive than existing motion-control technology, being able to track movements to 1/100th millimeter (0.01 mm) with no visible lag time. It also tracks hands and all 10 fingers at 290 frames per second. Its field of view, technically speaking, is 8 cubic feet of interactive, three-dimensional space: 2 feet above the controller, by 2 feet wide on each side (150º angle), by 2 feet deep on each side (120º angle). Aside this, the Leap Motion API measures physical quantities with the following units: distance (millimeters), time (microseconds unless otherwise noted), speed (millimeters/second) and angle (radians).

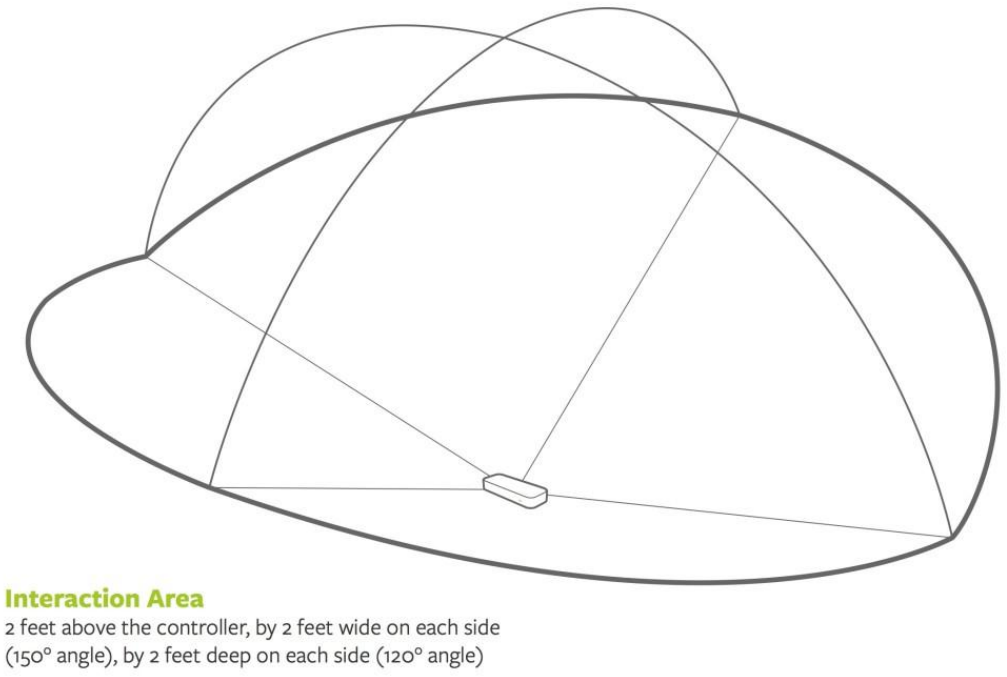


Figure 2.3: Leap Motion Controller Interaction Area

See Appendix A for the Leap Motion Controller technical specifications and the recommended system requirements.

The Novint Falcon

The Novint Falcon (Figure 1.4.a) is a USB haptic device which requires a power supply and that has removable handles, or grips, that the user holds onto to control it. As the user moves the grip in three dimensions (right-left (x-axis) and forwards-backwards (y-axis), like a mouse, but also up-down (z-axis), unlike a mouse), the Falcon's software keeps track of where the grip is moved and creates forces that a user can feel, by sending currents to the motors in the device. The Falcon's sensors can keep track of the handle's position to sub-millimeter resolution, and the motors are updated 1000 times per second (1 kHz), giving a realistic sense of touch.

The Falcon (Figure 2.4) is in essence a consumer robot which consists of its grip connected via three arms to a roughly conical body, which sits on a U-shaped base. Each of the three arms moves in and out of the Falcon's body. The default grip is a small spherical grip with 4 buttons on the top. The buttons are the Novint Logo for the primary button (which is similar to an 'N'), an upside down triangle (similar to a 'V'), a lightning bolt (similar to an 'N'), and a plus (similar to a 'T'), which collectively make the letters 'NVNT', the consonants in Novint's name and its ticker symbol as a public company. At the front flattened point of the Falcon's conical housing is a Novint Falcon logo that lights up in different colors to indicate the state of the device.

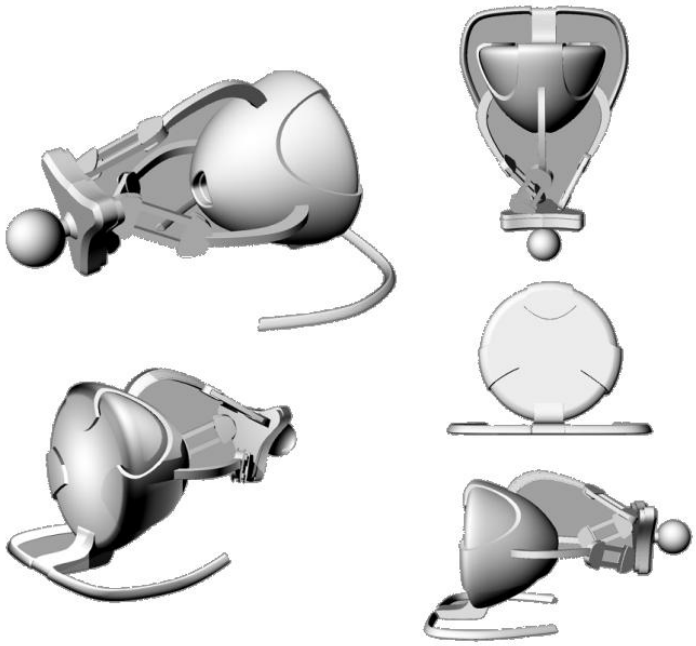


Figure 2.4: Novint Falcon 5-Views

The body contains 3 motors, each attached to one of the Falcon's arms by a cable that is wrapped around a capstan on the motor. As each of the 3 arms moves, an optical sensor attached to each motor keeps track of the movements of the arm. A mathematical function called a Jacobian is then used to determine the position of a three-dimensional cursor in Cartesian coordinates based on the

positions of the arms. The position of that haptic cursor is therefore controlled by the Falcon's movements, and is used by the Falcon's software to determine the forces to be applied to the user. Currents are sent to the motors at the 1 kHz servo rate to present the user with an accurate sense of touch. In this way, a force can be applied to the grip in any direction, up to the maximum force (over 2 pounds of force), every 1/1000 of a second (1 msec). Plus, the base has extra weight in order to provide necessary stability when larger haptic forces are triggered.

See Appendix B for the Novint Falcon Technical Specifications and the Recommended System Requirements.

2.2 Software

Leap Motion Airspace Home

The Leap Motion Software is easily downloadable from the Leap Motion website [SETL]. It provides the user with a robust platform called Leap Motion Airspace Home (Figure 2.5) where some applications to get started with the Leap Motion Controller are provided. There is where you also can have direct access to all your Leap Motion application downloaded from the Leap Motion Airspace Store, which characteristics will be shown in the next section.

Figure 2.5: Leap Motion Airspace Home

Once the Leap Motion Software is installed on the computer, the way to get started couldn't be easier. It just takes 3 steps and few seconds.

(a) Step 1 (b) Step 2 (c) Step 3

Figure 2.6: Leap Motion Getting started

Step nº1: Remove the sticker from the top of the controller. Step nº2: Connect it to your computer (use the included USB cable in the box. The bright side of the controller up and green sides toward you). Step nº3: Put yourself at ease, place the controller in front of you and make room to put your elbow.

To access the Leap Motion Airspace Home and start interacting with the whole Leap Motion world, see Appendix C.

Leap Motion Diagnostic Visualizer

Within the Leap Motion Software there is an application, not included in the Airspace Home, called Leap Motion Diagnostic Visualizer [DVZ], which lets you view motion tracking data generated by the Leap Motion Controller.

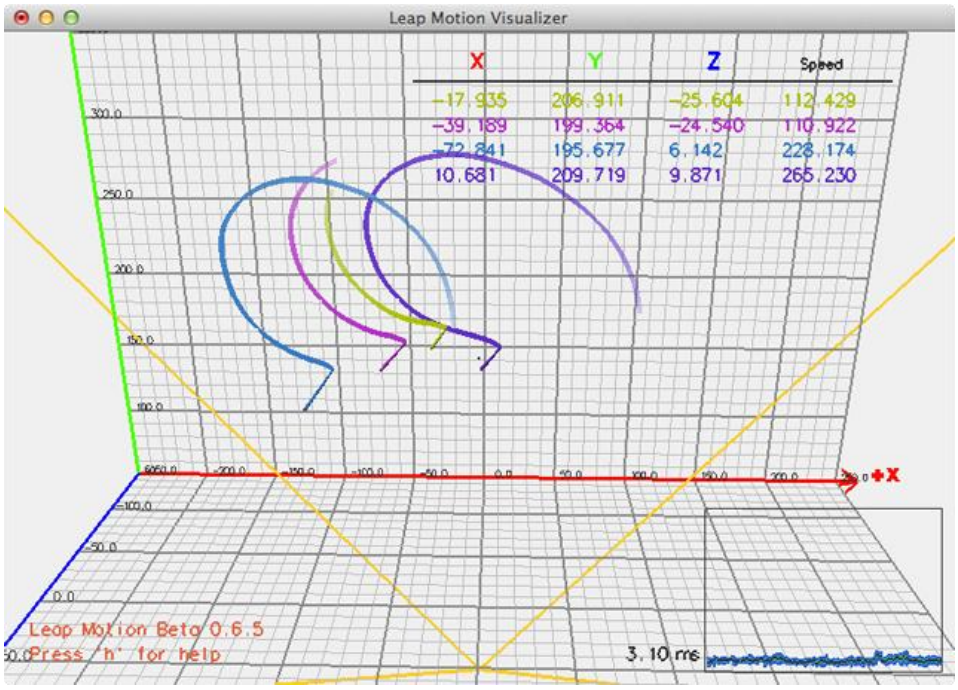


Figure 2.7: Leap Motion Diagnostic Visualizer

The Visualizer application displays a variety of tracking data provided by the Leap Motion API and is a good way to get a feel for the data produced by the Leap.

The Visualizer can be staarted from the Troubleshooting page of the Leap Motion Control Panel [CTL] by clicking the *Diagnostic Visualizer* button.

In this way, when you move your hands in the Leap Motion controller's field of view, you should see your fingers represented as colored arrows with trails following your finger tip locations, and much more related information.

Leap Motion Airspace Store

The success of Leap lies primarily in the wealth of compatible applications. Leap Motion has set up a wide open 1 year program to developers. 12,000 of them have received a housing in advance with the mission to develop new software. The Store counts with a few hundred of applications at launch, with prices ranging from free (for 15 games and utilities, including useful to control the interface of Windows 8, surfing, etc) to 99$. Most apps are priced. Being 1.99$ the average. Normally there are no free demos, and each one weighs between about ten to several hundred MB.

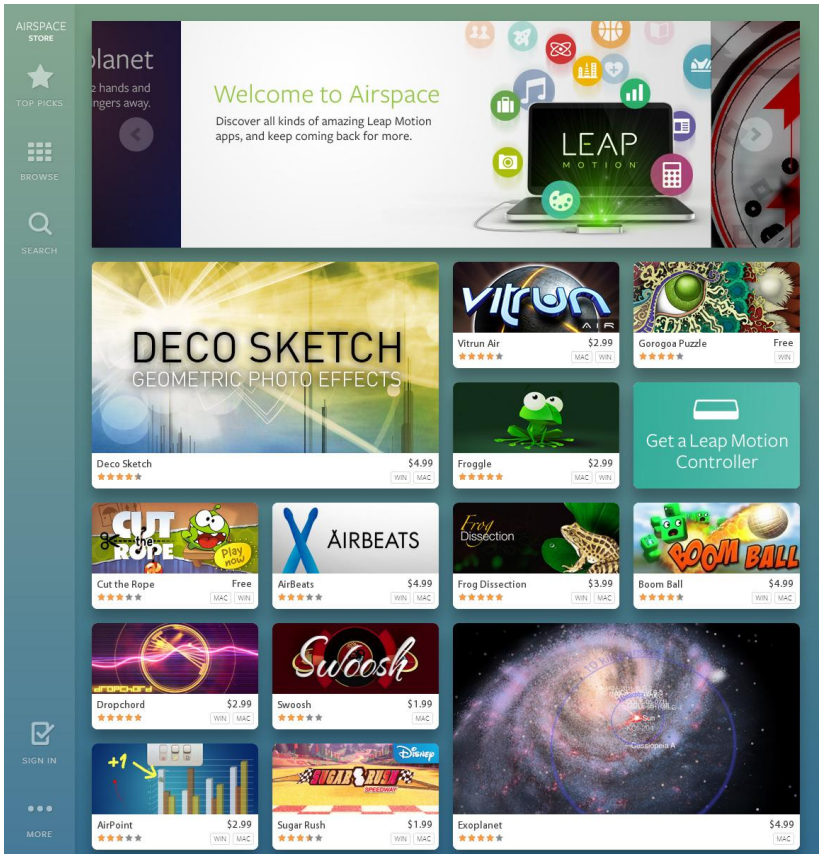


Figure 2.8: Leap Motion Airspace Store

The Airspace Store is divided into seven categories, which can actually be summarized in four main families, given the overlap: graphic and musical creation, edutainment, games and experimental. One could also mention the productivity category and utilities but it is actually a shell (almost) empty for the moment at least. Some of the application that can be found range from the well-known *Cut the rope, Fruit Ninja or Google Earth,* to others totally new which present a whole world of possibilities of this new technology.

The Airspace Store is directly accessible from the Airspace Home of Leap Motion.

Leap Motion Software Development Kit (SDK)

The Leap Motion Developer SDK [SDKL] contains the necessary libraries to get started with integrating the Leap Motion Controller into any project. You will find pre-compiled libraries, documentation, and some simple examples to get you started. It counts with versions for Windows, OSX and Linux, as well as with a very complete API Reference which provides details on all the classes which make up the Leap Motion API. This supports several programming languages such us JavaScript, C#, C++, Java, Phyton or Objective-C.

The Leap Motion SDK also includes 'Hello World'' example which, in a very easy and intuitive way, demonstrates how to use the Leap API to listen for frame events dispatched by the Leap and how to access the hand and finger data in each frame. The application is a small command-line program that prints information about detected hands and fingers to standard output, and it is contained in a single file, `Sample.cpp`. [TUTL]

All this and much more supplementary material can be found in the developer section [DEV] that Leap Motion has dedicated to provide all its developers with all the necessary information to success in the developing process of their Leap applications.

Leap Motion counts as well with a Leap Motion Community where its developers can interact and freely share its own experiences when interacting with the Leap Controller [COM], suggest questions, etc. A Leap Motion Blog [BLG] is also up to Leap developers where all the actuality and news regarding to Leap Motion technology and to the company are presented.

Appendix D fully describes the steps to follow in order to setup a project using the Leap Motion SDK.

Novint Falcon Bundled Software

Novint has bundled a variety of games with the Novint Falcon in a "Limited Edition Novint Falcon Bundle" which also includes a very useful tutorial. Running the tutorial is fully recommended as it highlights the best aspects of the Novint Falcon, its haptic feedback abilities. Various surfaces are simulated during the tutorial, ranging from sandpaper to ice to honey, each of which simulates the actual "feel" as the user is moving the handle over its surface. Moving the handle over sandpaper results in a grainy, rough haptic sensation, while the honey will produce a very sticky sensation, making it difficult to move the handle away. Furthermore, the tutorial highlights another haptic aspect with a solid cube which stops the handle from penetrating the cube once you contact the surface. Through these simple examples, it is easy to see how haptic feedback can drastically alter a users digital world perception when using the Novint Falcon.

The bundled games are "Newton's Monkey Business" and "The Feelin' It Sports Pack" which are accessed via a Steam-like interface called NVeNT (Figure 2.6) which is primarily aimed at a younger audience.

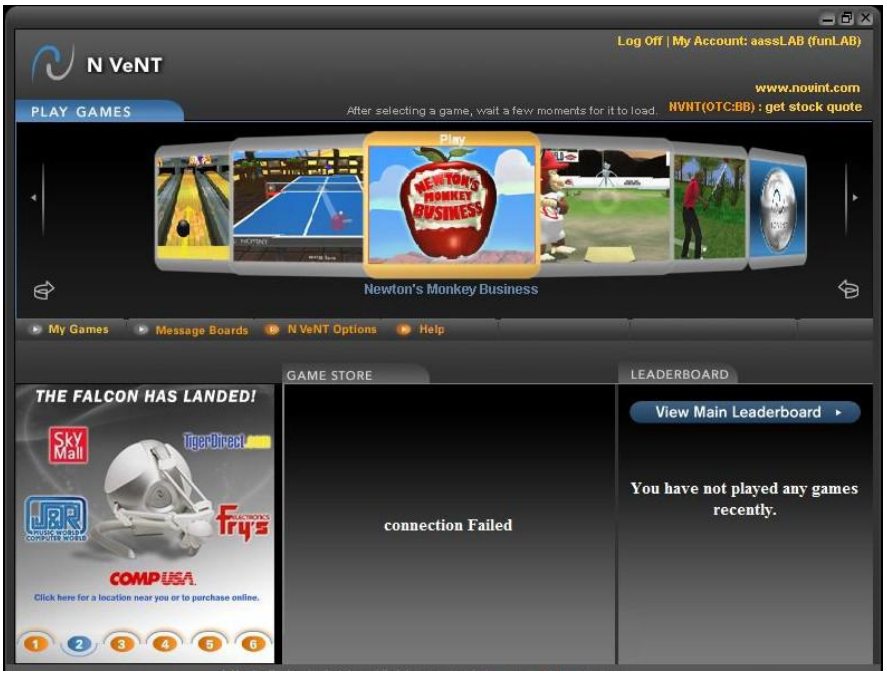


Figure 2.9: NVeNT Software

Hundreds of games can be purchased and accessed through the NVeNT interface. Once purchased, customers can use the bundled haptic responses or create, customize and use their own. Novint Falcon support exists for numerous top end gaming titles which includes Half-Life 2, Call of Duty, F.E.A.R., Left 4 Dead and most recently, Crysis 2. The user is able to customize the amount and type of feedback using Novints own customization tool, "F-Gen" which will produce a set of F-Gen Drivers.

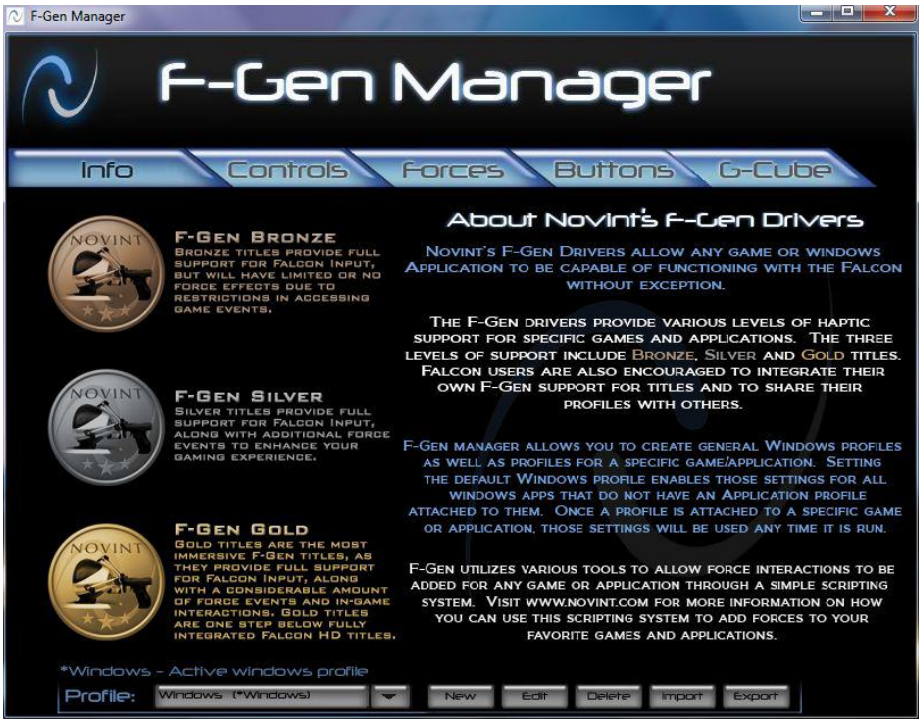


Figure 2.10: F-Gen

This mainly centers around the "feel" given for a particular attack or defense while playing the game of choice and despite Novint including a bundled haptics package, there is strong demand for customization among users. "The Falcon Army", Novints own forum group, provides a common ground where users can come together, share ideas, customizations and educate one another on haptics in general.

To use the Novint Falcon and complete the recommended tutorial, please see Appendix E for the General Setup instructions.

Squirrel

The simplest introduction to programming the Novint Falcon is to use a scripting language called "Squirrel" [SQR]. Squirrel is a high level imperative, objectoriented programming language, designed to be a light-weight scripting language that fits in the size, memory bandwidth, and real-time requirements of applications like video games. Developed by Alberto Demichelis in 2004, Squirrel has a syntax similar to C, C++, Java and has progressed to v3.0 with further developments planned for future releases. When using Squirrel, very little prior programming experience is required to successfully execute haptic commands with the Novint Falcon. Setting up Squirrel is a straight forward process and a brief setup guide is in Appendix F. Button states and keyboard mapping can be easily programmed to provide various levels of "feedback" to the user. Squirrel scripting is an excellent way to become familiar with the Novint Falcon and can be used to determine what is haptically possible with the Novint Falcon, thereby giving fresh ideas for the main project.

Novint Falcon Software Development Kit (SDK)

Once familiar with Squirrel, the next step is to delve into the Haptics Distribution Abstraction Layer (HDAL), which is provided with the Novint Falcon SDK. The HDAL allows near total control of the Novint Falcon but does require a solid background in C++ programming skills.

The Novint Falcon was released in 2007 on the Windows XP platform and is written using C++ in the Visual Studio 2005 (VS 2005) IDE. Windows XP has since been replaced by Windows 7 which works just as well. Although newer versions of VS are available, issues when trying to convert from VS 2005 to VS 2008 or VS 2011 do exist so it’s best to avoid newer versions of VS in order to not find this kind of troubles.

The HDAL uses OpenGL for graphical representations of the Novint Falcon cursor and for any objects it interacts with on the screen. Learning the basics of OpenGL is recommended but any advanced understanding is not necessary as OpenGLUT can be used due to its simplicity. OpenGLUT, an extension of the original GLUT [GLU1] project, making it easier to learn and explore OpenGL programming by offering a portable API so one can write a single OpenGL program that works across all OS platforms. GLUT [GLU2] is owned and copyrighted by Mark Kilgard who stopped working on the project in 1999. Since this time, alternatives to GLUT have surfaced with OpenGLUT being one of the better choices, in large part for its open source licensing.

Follow the steps in Appendix G to install the Novint Falcon SDK (HDAL). Appendix H describes what is needed in order to use OpenGLUT.

Novint Falcon SDK Basic Example

To ensure that the Novint Falcon SDK is installed correctly, it is recommended to run the included Basic example, but first ensure that the Novint Falcon is connected and powered on.

- Navigate to "C:/Program Files/Novint/HDAL_SDK_2.1.3/examples/Basic /vs2005/basic_opengl".
- Open the VS 2005 project file 'basic_opengl.vcproj' and run the project.

A new window will appear with a 3-Dimensional Cube in its center and the cursor will be represented by a sphere which is controlled by the Novint Falcon. The main purpose of this example is to show the haptic capabilities of the Novint Falcon where the sphere (cursor) is unable to penetrate any side of the solid cube. This is conveyed to the user by the abrupt stop of Novint Falcon movement once the sphere comes in contract with the cube. This holds true for each side of the solid cube.

Novint Falcon SDK Multi Example

A Multi Example is provided by the Novint Falcon SDK as well. It is very similar to the Basic example but this one focuses on the execution at the same time of several Novint Falcon hosted in the same computer. It shows the way to differenciate, initializate and control each of them separately. To run the Multi example:

- Navigate to "C:/Program Files/Novint/HDAL_SDK_2.1.3/examples/Multi".
- Open the VS 2005 project file 'multi.vcproj' and run the project.

As in the Basic example, a new window will appear with a 3-Dimensional Cube in its center but this time it will exists a cursor for each Novint Falcon connected to the computer (with a maximum of 4) represented as well by a sphere of different colours which is controlled by the each Novint Falcon. The behavior of this example is the same as the previous one with the big exception that now more than one Novint Falcon is handled.

Chapter 3
Developed System

3.1 Overview

This project has set out to use the Leap Motion controller as a control device for the haptic feedback device developed by MINT research group (Section 1.5), so that a continuous adaptation of the position, elevation and orientation of a tactile end-effector pad to the position of the user finger is achieved, and where no contact between the interaction pad and the user is considered. According to this (Section 1.6) the main motion to be controlled is the position of both Novint Falcon in order to, as the user's finger approaches to the tactil pad, it is already set with the right position, elevation and orientation, regarding to the virtual shape that is supposed to me rendered. All this means the behavior of the Novint Falcons must be totally precise at everytime. Moreover, the integration between these and the Leap Motion controller must allow us to have a real kinesthetic experience, that is a real sense of touch, when interacting with the robot, as well as being everything developed in real time. With this purpose an early 'Leap Motion + Falcon Novint haptic device' prototype has been developed, which consists of a platform where the Leap Motion controller is faced with the haptic shape rendering device in a way that the interaction area of both occupy the same space. In addition, the Leap Motion's interaction area has been programmed in a 2D picture format, with a given resolution in each direction (x and y), where the information of each pixel is the heigh the shape to render takes in that position. When the user finger comes into the interaction area of the Leap Motion controller, thanks to the way it's been programmed it is possible to predict the potential contact point of the user finger with the tactile pad in each moment, that is, the potential contact pixel of the shape to render. This allows us to continuously adapt the Stimtac to the user finger.

Several functions have been developed to initialize and handle both devices, the Leap Motion controller and the Novint Falcons, as well as to integrate both in the same platform explained above.

Taking into account that all the developed work has been carried out with the early prototype of the haptic device developed somo time ago by MINT lab, but that a new more complete prototype will be ready by the next year, as it's been explained in previous sections (Section 1.6), all the developed system has been done allowing a quick, intuitive and easy adaptation of the whole system to this new version when it is ready.

3.2 Simulated Surface

The very last goal of this project is to be able, by the interaction with the developed haptic feedback device, to render the shape (and texture) of a virtual 3D surface.

The way this 3D surface to be simulated is defined is as a 2D picture (x-dimension and y-dimension) with a certain resolution in each one (by default the same one), where the data stored in each pixel is the height that the surface takes at that point, getting therefore the 3D shape that will be simulated by the Novint Falcons device. Thus, a grayscale 2D picture could be defined, where the level of grey is the eight of the 3D surface at each pixel.

In our case, since we are working with an early prototype of the developed device the chosen surface to carry out all the experiments that have taken place is a very simple convex surface (parabole).

The way all this it's been programmed is by defining a *class surface* where the method called `defineSurface` is the most important one since it is where the picture itself is defined as a bidimensional array where each dimension corresponds with the two dimensions of the picture we are designing, and where the data of each position of the array contains the height of the shape at that pixel of the picture. The input parameters of the method define the resolution of the picture in both x and y dimension.

The greatest advantage of the chosen way to define de virtual surface is that any given picture with the same format can be directly simulated without the necessity of including any change.

See Apendix I for a better appreciation and understanding of the way the virtual surface is defined.

3.3 Leap Motion

Within all the possibilities offered by the Leap Motion technology only its most basic and simplest functionality is implemented in this project, that is, its capability to detect and track hands and fingers placed within its field of view. The Leap captures this data one *frame* at a time. Very useful data that we'll access by using the Leap API to achieve the continuous adaptation of the end-effector pad that we are looking for.

Keeping this in mind, our application must listen at every time for frame events dispatched by the Leap, and access the hand and finger data in each frame.

Creating a Controller object

The *`Controller`* class provides the main interface between the Leap and our application. When you create a Controller object, it connects to the Leap software running on the computer and makes hand tracking data available through *`Frame`* objects. You can access these `Frame` objects by instantiating a `Controller` object and calling the `Controller::frame` function.

If your application has a natural update loop or frame rate, then you can call `Controller::frame` as part of this update. Otherwise, you can add a *`Listener`* object to the controller, that is exactly what we do in this project. The controller object invokes the callback functions defined in our `Listener` subclass whenever a new frame of tracking data is available.

Our application creates a `Controller` object in its `main` function and adds an instance of a `Listener` subclass to the `Controller` with the `Controller::addListener` function.

However this is not enough yet. We also need to create a subclass of *Listener* named *SampleListener*. The Listener subclass defines callback functions which the controller calls when a Leap event occurs, including when a frame of tracking data is ready.

Subclassing the Listener class

The defined *Listener* subclass, SampleListener, implements callback functions to handle events dispatched by the Leap. The events include are:

- onInit — dispatched once, when the controller to which the listener is registered is initialized.
- onConnect — dispatched when the controller connects to the Leap and is ready to begin sending frames of motion tracking data.
- onDisconnect — dispatched if the controller disconnects from the Leap Motion device(for example, if you unplug the Leap device or shut down the Leap software).
- onExit — dispatched to a listener when it is removed from a controller.
- onFrame — dispatched when a new frame of motion tracking data is available.

For four lifecycle event callbacks, onInit, onDisconnect, onConnect and onExit our application simply prints a message to standard output. For the onFrame event, the listener callback does a bit more work. When the controller calls onFrame, the function gets the latest *Frame* of motion tracking data (useful data) and use this data in the way we implement to achieve the pursued goal.

Getting a Frame of data

The *Controller* calls the onFrame callback function when the Leap generates a new frame of motion tracking data. You can access the new data by calling the Controller::frame function, which returns the newest *Frame* object. (A reference to the Controller object is passed to the callback as a parameter.) A Frame object contains an ID and a list containing a *Hand* object for each physical hand in view.

Our application's onFrame implementation gets the most recent Frame object from the controller, retrieves the list of Hand objects from the Frame (normally only one because the application is designed to be in this way) and from the Hand object retrieves the list of Finger objects (normally only one as well because the application is designed to be used just with one finger). In case it detects more than one finger it just takes the first one of the list. Later the application uses the information of the position of this finger (user finger) to adapt the tactile pad to its position in a way that will be explained in detail in the next sections.

Check out Appendix I to access the source code.

Defining the Virtual Surface

The information of the user finger by its own is totally useless. This data is totally necessary but with regard to the virtual surface that is supposed to be rendered. Leap Motion technology allows us to carry out this task by defining its area of interaction.

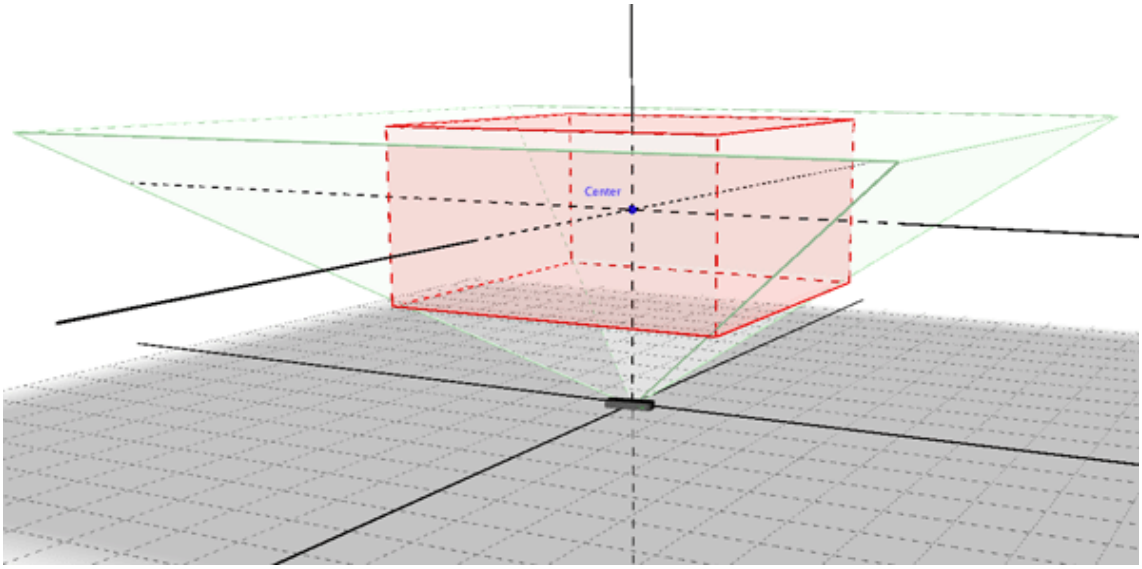


Figure 3.1: Defining Leap Motion Interaction Area

In this way, the Leap's interaction area is limited to the ROM (Range of Motion) of the haptic device, and at the same time is programmed with the virtual surface defined before. Thus, the interaction area of both devices, the Leap and the haptic one, are the same in x and y dimension and they correspond with the picture of the virtual surface conceived to be rendered. Once defined over the Leap software, the picture is characterized by another parameter called *RESOLUTION* which defines the real resolution of the virtual picture at the time to be rendered by the haptic device. Actually, the value of this parameter is the size, in x and y dimension, of the pixel of the virtual picture. With this value and the total size of the picture, defined by the ROM, the final resolution of the virtual picture can be easily calculated. According to this, a high value of this parameter means a low resolution, and a low one a good resolution, and therefore a good sense of touch when interacting with the haptic device. Thanks to this is possible to refer, at any time, the position of the user finger to a pixel of the picture, and therefore of the surface, and access to the data stored in it and use it to define the position of the Novint Falcons in order to properly simulate the defined virtual shape by the continuous adaptation of the end-effector tactil pad, in position, elevation and orientation.

Check out Appendix I to access the source code.

3.4 Tactile Plate Position, Elevation and Orientation

According to the previous section, once the potential contact pixel of the surface's picture is been predicted, by the measured position of the user finger within the Leap Motion workspace, it is possible to access the data stored within it, as well as to the data of the next and previous pixel, in order to properly set the tactile pad at any moment.

In the context of the whole 2D picture (x and y-dimension) each pixel has directly associated an x-coordenate and a y-coordenate in the Leap's coordinate system (Transformation between Leap's coordinate system and Novint Falcon's coordinate system will be explained in the next section) so, in this way the position of the tactile plate would be done.

Accessing to the data of the pixel, which is the eight the surface takes at that point of the picture, the elevation of the plate would be set as well. However, it remains to set the right orientation of the plate in order to properly render the virtual surface that is being simulated, and thus, achieve a real haptic interaction with the device. In this way, for each point of contact with the surface is necessary to be able to continuously simulate the tangent plane to that point so that a real surface sensation can be rendered [3] [4].

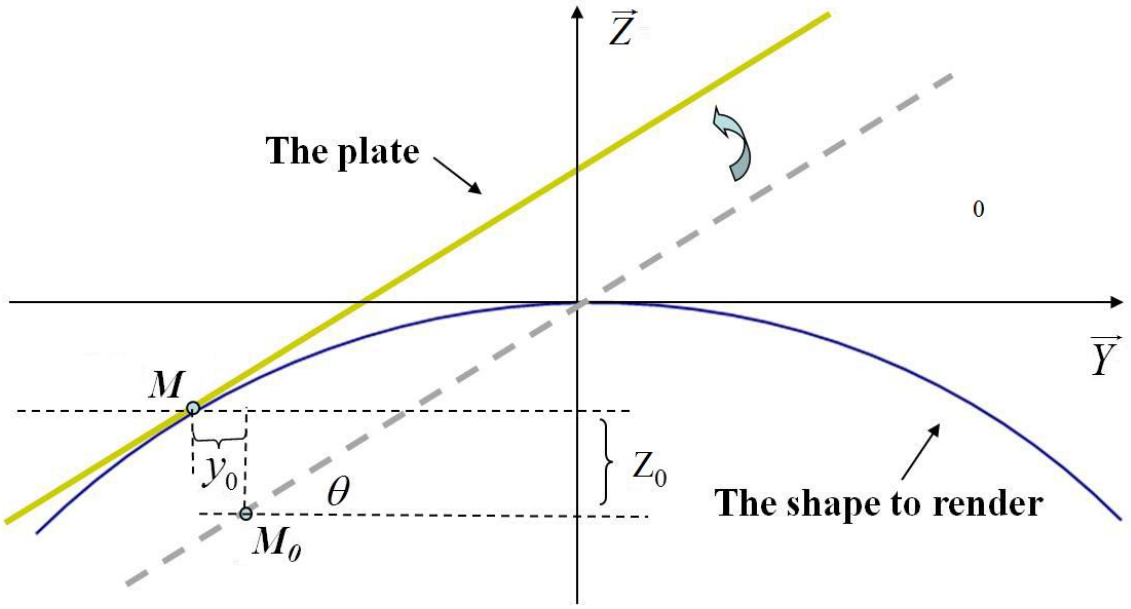


Figure 3.2: Principle of Shape Rendering

Since the contact pixel point is known for us, the next pixel in both dimensions (x and y) are known as well, which in fact represent 3 points of the same plane (the plane we are looking for), plane easily calculated from them. As it's been explained in previous sections, because of the physical structure of the prototype used in this project is not possible to orientate the plate in the x-direction, so in this project will only be calculated the orientation of the plate in the y-direcction, but following the same principle explained below the orientation of the plate could be achieve in the new prototype for the x-direction as well.

According to what's been said, once we know the potential contact pixel (M) we know as well the next one (M_0). With this information we know the heigh of the surface at both points, and therefore the difference between those values (z_0), as well as the distance between them, which is the picture resolution in this direction (y_0). Thus, the orientation of the plate (θ, in degrees) can be easily calculated as follows:

$$\theta\ (degrees) = arctg\ (\frac{z0}{y0})$$

At this point, the position, elevation and orientation necessary to simulate the surface at the predicted contact pixel are known. The final point to succeed is to be able to properly set both Novint Falcons so that the end-effector plate respects these theoretically calculated values: position (x-y-direction, in mm), elevation (z-direction, in mm) and orientation (θ, in degrees).

Check out Appendix I to access the source code.

3.5 Workspace Matching

One of the challenges in this project was to match the workspace of the Leap Motion controller with the workspace of the Novint Falcons used to build the haptic interaction device because much of the success of the project resided in this point.

As it's been said in previous sections both workspace match in 2D (x and y dimension) however it shouldn't be forgotten that with regard to the Leap Motion Controller it is done in a virtual way (software part), while when it comes to the Novint Falcons it means a real physical ROM.

To achieve the instantaneous and continuous control of the plate position, the information given from the Leap Motion Controller and based on the movement law, that is, position, elevation and orientation (x, y, z, θ), has to be translated into references (x1, y1, z1) and (x2, y2, z2) for the two Falcon systems (Figure 3.3).

Novint Falcons' position (x and y coordinate) and StimTac's position are the same, therefore the real challenge is the elevation and orientation achievement, and more precisely the orientation achievement for several reasons explained below.

Novint Falcons' position (x, y and z) is well known at any time (Novint SDK includes functions that bring us this information), which means that the height difference between both is easily calculated. The degree of orientation is also known according to the previous calculations. All this information allows us to simulate the orientation of the plate as the next figures shows:

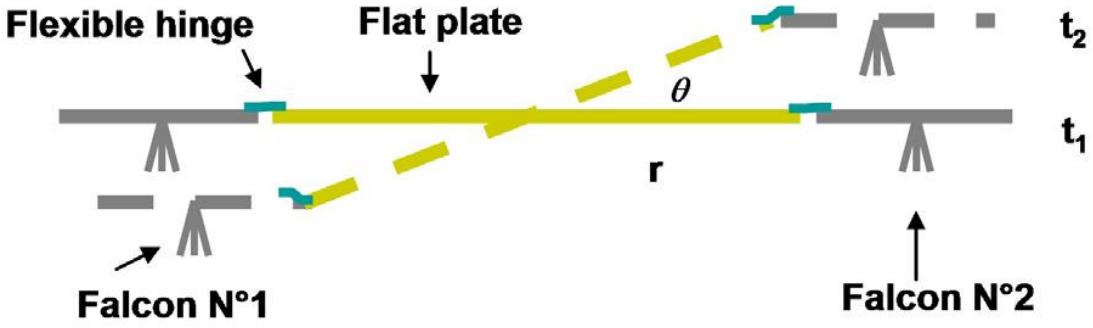


Figure 3.3: Control organization with the two "Falcon" systems

Being *r* the size of the flat plate and *z1, z2* the eight of Falcon 1 and Falcon 2 respectively, the orientation θ of the plate can be calculated as:

$$\theta\ (degrees) = arctg\ (\frac{z2 - z1}{r})$$

Since θ and r are know we have 2 degrees of freedom to solve the equation. This allows us to set the height of one of the Falcons (the chosen height is 0, the minimum) and then calculate the value of the height the other one must take to achieve the right orientation. In this way it must be noticed that regarding to the value of the orientation we can know if the plane we are calculating is upward (θ > 0º, that is, z2 > z1, which means that next pixel's heigh > contact pixel's height), parallel to the floor (θ = 0º, z2 = z1, next pixel's heigh = contact pixel's height) or downward (θ < 0º, that is, z2 < z1, which means that next pixel's heigh < contact pixel's height). Thus, in the first case (θ > 0º) Falcon 1 is set with elevation 0 and Falcon 2 elevation is calculated according to the expression above and in the opposite case (θ < 0º) Falcon 2 is set with elevation 0 and Falcon 1 elevation is the calculated one. When the orientation is null (θ = 0º) then z2 = z1 = elevation previously calculated.

Finally, once the orientation of the plate has been achieved, its real elevation is calculated as the mean of the elevation of the Falcons. In this way, by comparison with the elevation previously theoretically calculated it is possible to adjust the final elevation of the Falcons so that keeping the orientation calculated, the pad's elevation is properly set.

According to this, the plane we were looking, defined by its position, elevation and orientation would be successfully achieved.

Check out Appendix I to access the source code.

3.6 Coordinate System

Due to the utilization of two different technologies with different cartesian coordenate system along the next lines the project's coordinate system is clarified regarding to the physical structure finally decided to implement the platform to integrate the Leap Motion Controller with the haptic decide designed by the MINT research group.

Leap Motion Coordenate System

The Leap Motion system employs a right-handed Cartesian coordinate system. The origin is centered at the top of the Leap Motion Controller. The x- and z-axes lie in the horizontal plane, with the x-axis running parallel to the long edge of the device. The y-axis is vertical, with positive values increasing upwards (in contrast to the downward orientation of most computer graphics coordinate systems). The z-axis has positive values increasing toward the user.

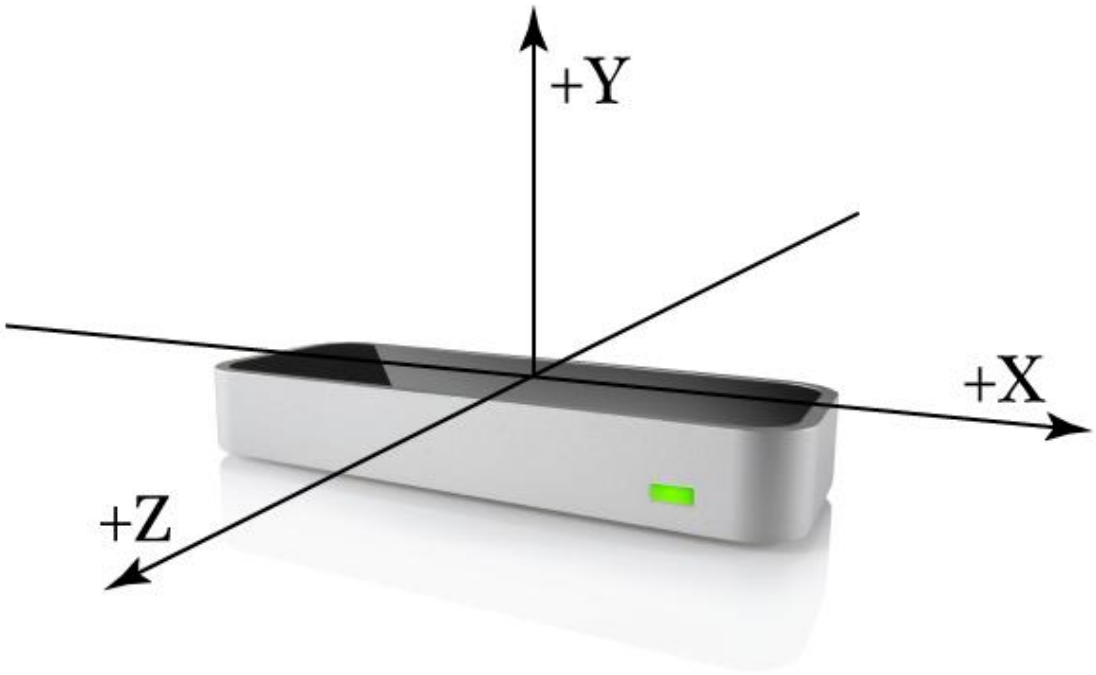


Figure 3.4: The Leap Motion right-handed coordinate system

It must be noticed that the Leap Motion's coordinate system doesn't take negative values for the y-axis due to its physical structure (the interaction area of the Leap Motion Controller just exists above it and not below (Section 2.1)).

Novint Falcon Coordenate System

Access to the position of each Falcon, as well as to the respective axis of each one is handled by an array of coordenates within the accompanying Novint Falcon SDK. This makes possible to find the actual range of movement of each axis (Table 3.1 and 3.2) and assign a safe range of motion (ROM) (Table 3.3), the same for both Novint Falcon. This is done because although the ranges of movement are similar for moth Falcon, actually are not exactly the same, and also because of in the limits these ranges, forces cannot be properly applied to the Falcons and in consequence they can't be right controlled. This is easily noticed because at these points Falcon's arms start to vibrate without control.

	Max	Min	Range
x-axis	6.3834 (right)	-5.9277 (left)	12.3111
y-axis	6.1492 (up)	-5.9999 (down)	12.1491
z-axis	3.3914 (out)	-6.2253 (in)	9.6167

Table 3.1: Actual Limits and Range (cm) for Novint Falcon Nº1

	Max	Min	Range
x-axis	6.5055 (right)	-5.9196 (left)	12.4251
y-axis	6.2756 (up)	-5.9564 (down)	12.1232
z-axis	3.4242 (out)	-6.1092 (in)	9.5334

Table 3.2: Actual Limits and Range (cm) for Novint Falcon Nº2

	Max	Min	Range
x-axis	5 (right)	-5. (left)	10
y-axis	5 (up)	-5 (down)	10
z-axis	-0.2 (out)	-6 (in)	7.9248

Table 3.3: Chosen Limits and Range (cm) for Novint Falcon handle

As can be noticed, the Novint Falcons' coordinate system follows the next organization: X-axis (right - left movement), Y-axis (up - down movement), and Z-axis (in - out movement). Where the positive senses, according to the prototype configuration, are *right*, *up* and *out*.

Moreover, according to the physical structure of the device (Section 2.1) the coordinate system of both Falcons match in the following way:

Novint Falcon Nº 1

Right (+x)	**Left (-x)**	**Up (+y)**	**Down (-y)**	**Out (+z)**	**In (-z)**	
	■					**Right (+x)**
■						**Left (-x)**
			■			**Up (+y)**
		■				**Down (-y)**
				■		**Out (+z)**
					■	**In (-z)**

Novint Falcon Nº 2

Table 3.4: Novint Falcons Matching Coordinate Systems

Project Coordenate System

According to the final developed system where the Leap Motion controller is faced to the haptic interaction device, as it's been explained in previous sections, with regard to a movement (mm) in the Leap Motion's coordinate system the match with the Falcons' system is as follows:

	+ x	- x	+ z	- z	+ y
Falcon 1	-y (down)	+ y (up)	+ x (right)	- x (left)	- z (in)
Falcon 2	+ y (up)	- y (down)	- x (left)	+ x (right)	- z (in)

Table 3.5: Project Coordenate System

3.7 Main Programme

The whole designed 'Leap Motion + Novint Falcon' integrated platform is handled and commanded by a c++ application developed with Visual Studio version 2005. The next lines present its general organization as well as its most outstanding parts.

Initialization

Once the system is started the first thing that takes place is the creation of the virtual surface to be rendered by the system. Immediately afterwards an initialization of the Novint Falcons (the Novint Falcon has its own initialization functions which have been supplied by the SDK) takes place. The main functions checked are the servos for each motor and encoder pair (3 in total) and that the device is "homed"[1]. If the device is not "homed" then the LED lights at its center will be red asking the user to move the handle in and out until the LED lights turn blue. This takes no more than a few seconds and it's something to be done every time we run our application after a long time of inactivity (1 hour minimum) so that the remaining functions can be properly initialized.

No hardware initialization necessary for the Leap Motion controller. The software of Leap Motion allows to its controller to be ready to be use since it is plugged into the computer by its usb-connector. The 'real initialization' of the Leap consists of the creation of a *controller object* and a *listener object* which allow its tracking functionality.

This type of initialization allows for a fluid and proper start so that the user can immediately begin using the devices as intended.

Application Core

The heart of the developed application resides in a never-ending loop which starts when the application is run and just ends up (non unexpected events taken into account) when the 'enter' key of the keyboard which host the whole system is pushed or when any button of any of the Novint Falcons of the haptic device is pushed as well.

Within this loop take place the hand tracking activity by the Leap Motion Controller and the accessing, treatment and computing data work, explained along the previous sections.

Over this tracking and computing loop, is implemented in background a fast position control loop already provided by the Falcon system which is originally focused on force feedback rendering. This loop is responsible of the control of both Novint Falcons' position, and with the data provided by the tracking loop, continuously adapts them so that the shape rendering goal of this project is achieved.

See Apendix I for a best and full understanding of the application. The whole source code can be found fully and accurately comented in this section.

Chapter 4
Experiments

4.1 Overview

Numerous experiments were performed before the final integration of the Novint Falcon and the Leap Motion controller could be completed. These have been highlighted in the text that follows.

4.2 Initial Tests

Within this group of experiments are included those carried out in the early stages of the project with the Leap Motion Controller and the Novint Falcon haptic device, both separately and together, in order to, on one hand become familiar with these technologies and devices, and on the other hand to verify the viability of the project.

Novint Falcon

There are two methods to program the Novint Falcon, Squirrel scripting (beginner) [SQR] and the SDK (advanced). The SDK offers many advantages over Squirrel when it comes to getting an accurate and complete control of the Novint Falcon, reason why Squirrel scripting was not used along this project and only the SDK was implemented.

The Novint Falcon SDK (HDAL) [SDKL] is an integral part of this project and requires a solid understanding of C++ to be fully understood. Implementing the SDK, adding haptic forces and integrating it with the Leap Motion SDK required a large part of the available project time frame. The gains made in flexibility come at the cost of a required higher level of C++ understanding. The following experiments were realized during this phase:

- Run Novint Falcon Test application to check the basic operation of the device as well as its proper installation to the computer.
- Run Novint Falcon Tutorial to get familiarized with the Novint Falcon haptic possibilities.
- Run Novint Falcon Basic and Multi example in order to experience a real application and GUI developed with the Novint Falcon SDK.
- Get x, y and z-axis position of each Novint Falcon.

- Measurement of the Range of Movement (ROM) of each Novint Falcon.
- Limitation of the ROM to a safety Range of Movement.
- Set each Novint Falcon in a given position (x, y and z-coordante).
- Study of the force applied to by the Novint Falcon motors in order to set them in a given position (x, y and z-coordante).
- Limitation in the maximum forces applied by the Novint Falcon motors as a security measure.
- Match both Novint Falcon coordinate systems so that the end-effector pad could be set in a given position (x, y and z-coordante).
- Add button functionality to all button of both Novint Falcon to quit he main program.

Leap Motion

The Leap Motion SDK brings to Leap developers all tools and functions to control with a total precision the main functionalities of the Leap Motion Controller, as well as the necessary background to develop new ones. The experiments completed during this phase include:

- Run some of the application of the Leap Motion AirSpace in order to get a first contact with the Leap Motion technologie and its hardware.
- Run the Leap Motion Sample example to get familiarized with the Leap Motion software and functionalities.
- Get the number of hands, fingers and tools within the interaction area of the Leap Mtion controller, as well as the position (x, y and z-coordante) of each one.
- Interaction with the Leap Motion Visualizer in order to get the best possible control of the device.
- Reduce the frame acquisition rate of the Leap Motion Controller.
- Modification of the size of the Leap Motion Controller interaction area.
- Interaction with the Leap Motion Visualizer in order to get the best possible control of the device.

4.3 System Tests

To successfully integrate an only system where Leap Motion and Novint Falcon technologies work together in order to achieve the main goal of this project the following experiments were carried out:

- Design of different shapes with the defined picture format in order to test the possibilities of the haptic device.

- Modification of the virtual picture resolution and study of the impact in the interaction sensation.

- Modification of the virtual surface size and study of the impact in the interaction sensation.

- Study of the position of the Leap Motion Controller with regard to the haptic device so that the the best physical integration of both device is achieved at the same time that the tracking functionality of the Leap Motion Controller doesn't see alterated its behavior.

- Initial default position of the haptic device when running the program.

- Study of the physical structure of the Novint Falcons haptic device in order to continuosly simulate the tangent plane to the virtual picture which allows us to render the simulated virtual surface.

- Study different organization regarding to achieve the best workspace matching between the Leap Motion Controller and Novint Falcons.

- Print on the scream some real information of the behavior of the device in order to study its response in real time.

- Print on texte files some real information of the behavior of the device in order to study its response in real time.

Obviously, the main functionality to be tested is the capability of the designed integrated platform to achieved the main goal of this project, that is, its capability to predict and adapt the position (x-direction and y-direction), the elevation (z-direction) and the orientation (degrees) of the end-effector tactil pad (Stimac) to the position of the user finger regarding to the simulated virtual shape that is supposed to be rendered.

In this line, some real measured were done, where the calculated theoretical position of the Stimtac in real time is compared with the real position achieved by this at each moment, thanks to the work of prediction and adaptation carried out by the designed Leap Motion + Novint Falcon platform. As it's been seen along this report, the position of the tactil pad is given by the control of two Novint Falcons. At the same time, each one of these has three robotic hands, and the way to control them is by defining the position of each one of these hands (in total six hands) at every moment. Thus, in order to check out the real success of the platform when it comes to predict and adapt the stimtac to the user finger, the exact position of each one of these 6 hands (3 of each Novint Falcon) has been measured and compared to the calculated theoric position they should place. These measurements have not been carried out just one random time, but 5, in order to get realistic data, with a duration of around 4-5 seconds. Therefore the given outcomes showed below are the result of the mean of those 5 measures. Plus, should be noticed that the following graphics represent the position of each hand of each Novint Falcon in meters (y-axis), and the x-axis is represented in samples (the samples achieved by the system in those measurements of 4-5 seconds), where the theoric behavior of the hands is showed in orange colour, and the real one in blue.

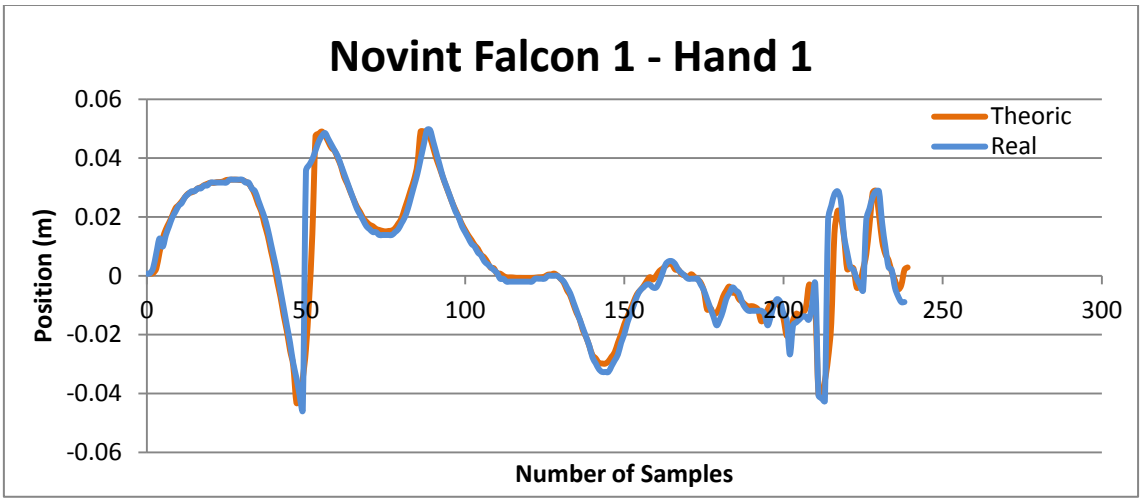


Figure 4.1: Novint Falcon 1 – Hand 1 measure

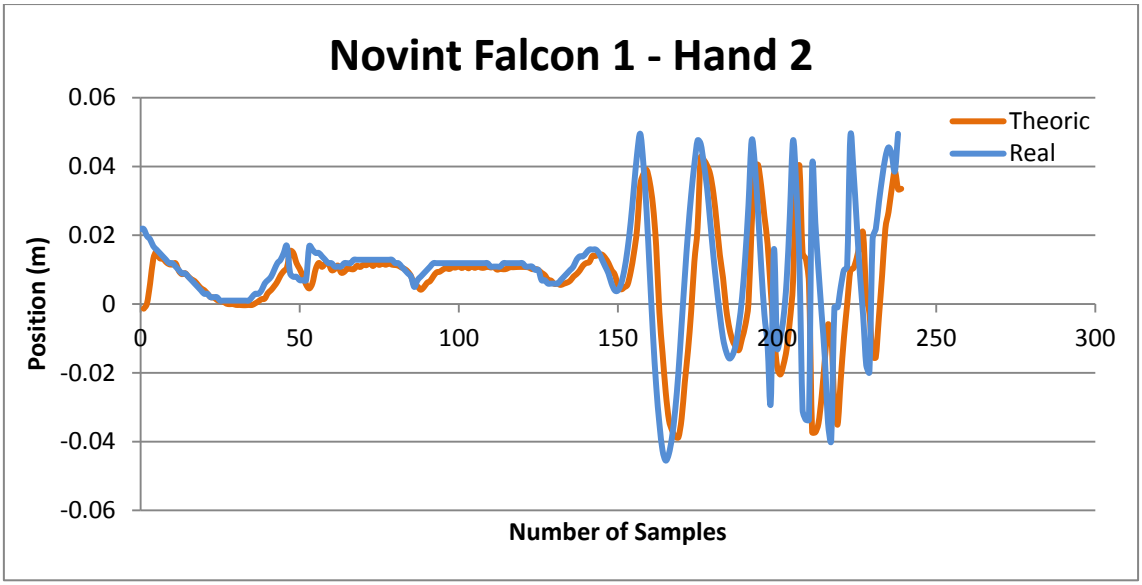


Figure 4.2: Novint Falcon 1 – Hand 2 measure

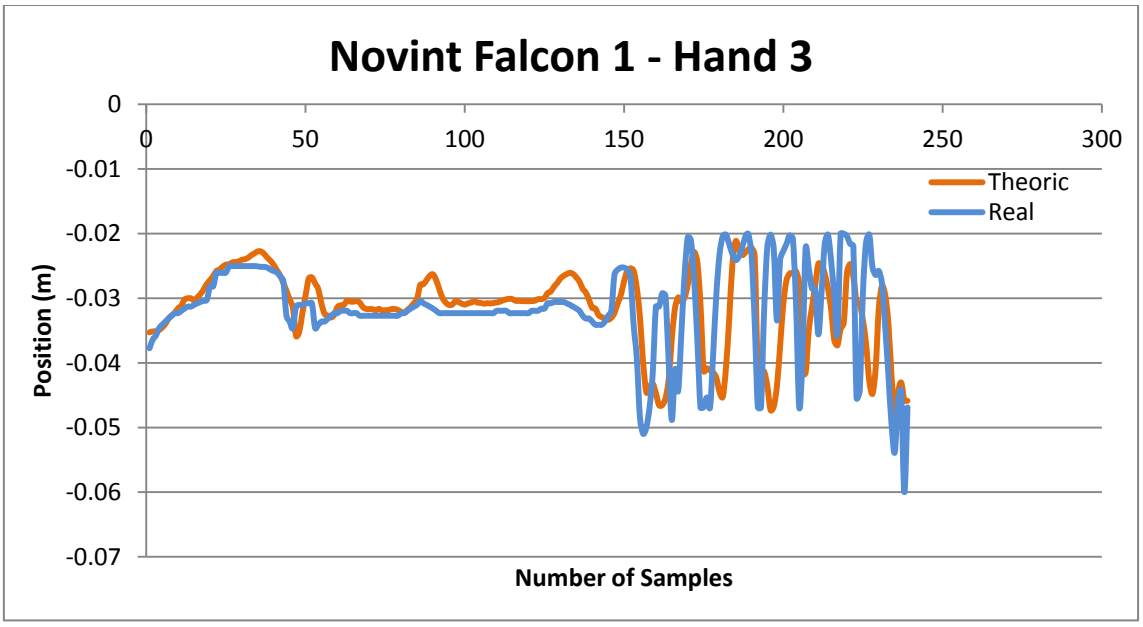


Figure 4.3: Novint Falcon 1 – Hand 3 measure

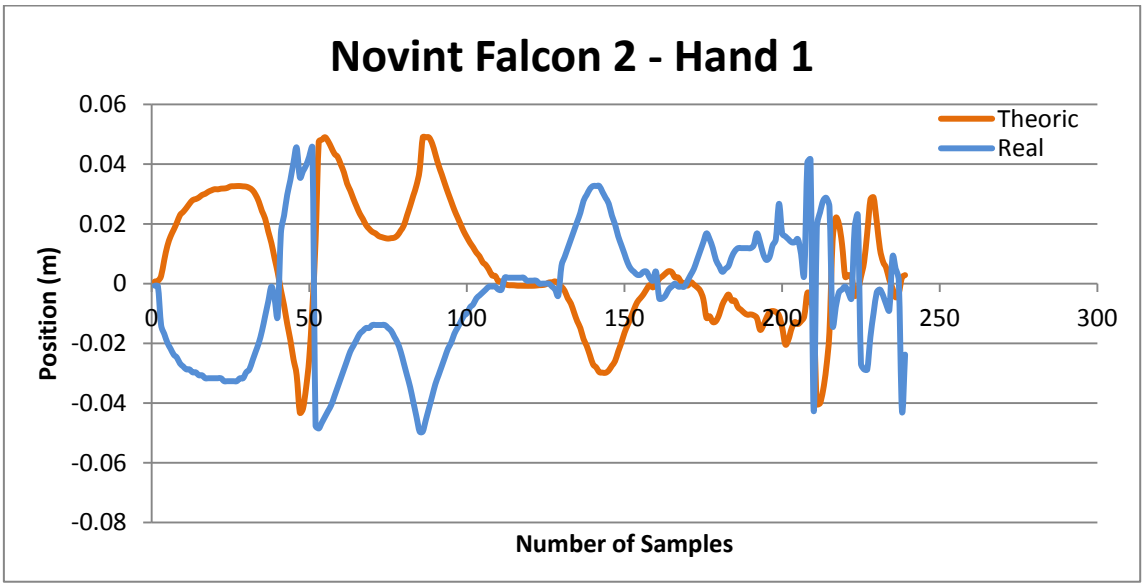


Figure 4.4: Novint Falcon 2 – Hand 1 measure

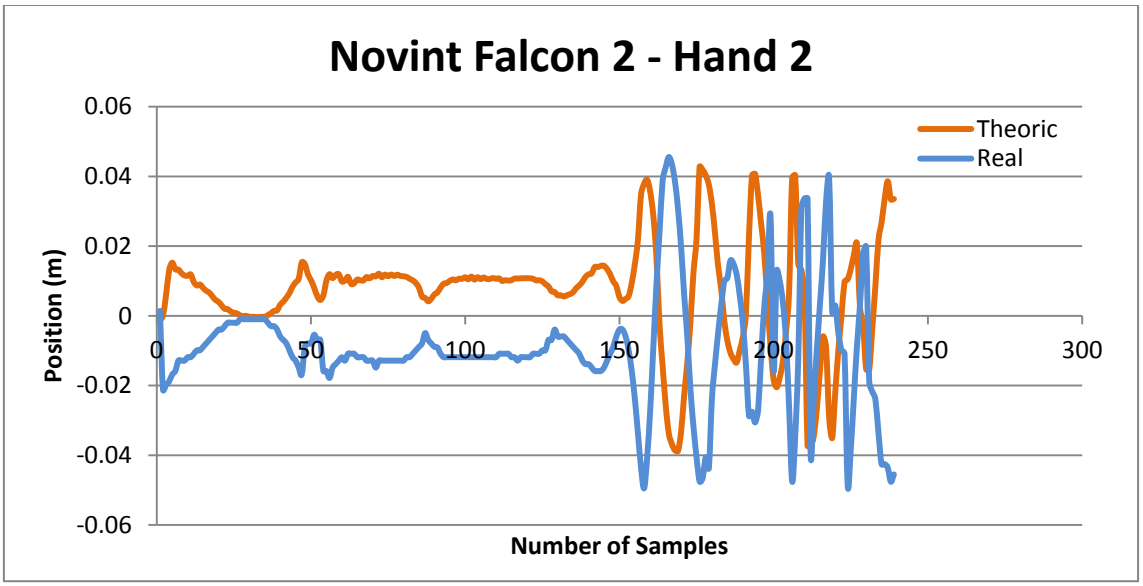


Figure 4.5: Novint Falcon 2 – Hand 2 measure

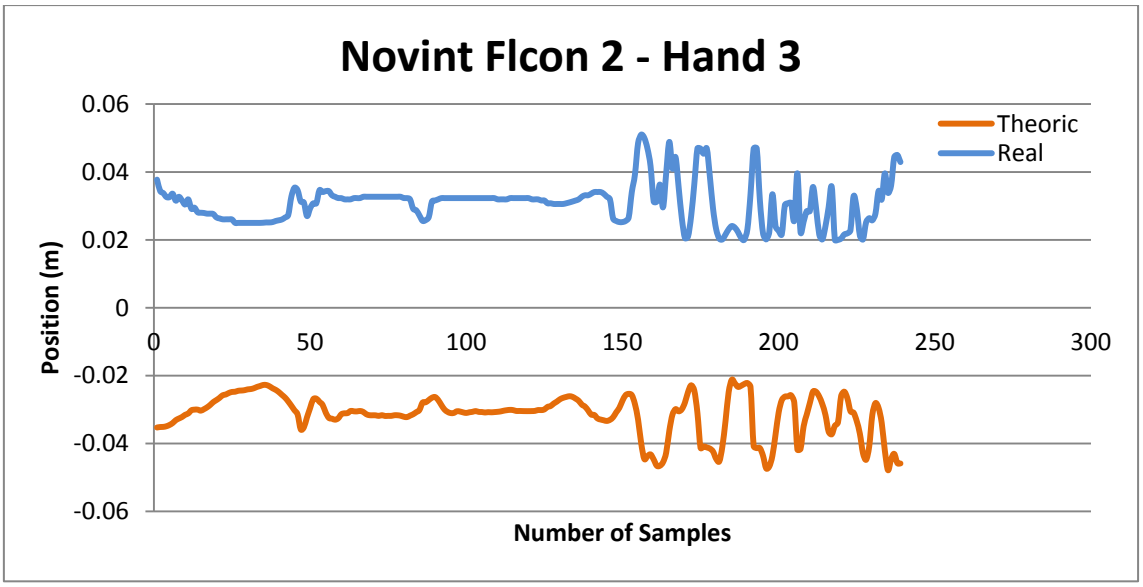


Figure 4.6: Novint Falcon 2 – Hand 3 measure

In real terms, the mean error of each falcon, measured as the diference between the theoric and the real position of each of its hands for each sample and divided by the total number of samples, and then as the mean error of each hand divided by three that is the number of hands of each device is: 0.00438067 m for the first Novint Falcon and 0.00516276 m for the second one.

Going a little bit beyond, it means that the mean error when it comes to predict and adapt the position of the tactil pad to the user finger in real time, with regard to the virtual shape to be rendered, is about 4 and 5 milimiters, result that can be considered as good since this is the first approach to integrate the Leap Motion with the Novint Falcon haptic device designed by MINT time some time ago.

This can be easily seen looking at the measurements of the hand 1 and 2 of both Novint Falcon. These figures are the ones which best show this. Overall in the case of Novint Falcon 1, since for the second one it seems less intuitive, where for its first hand, for example, the theoretical and the real position almost perfectly

match in some sections. When it comes to the Novint Falcon 2 the match between the theoric and the real case apparently might seem worse due to its x-axis simetric condition with regard to the Novint Falcon 1 position and to the theoric position. This happens basically due to the way the haptic device is designed. As it's been said in previous sections, the haptic device designed by MINT team consists of two faced Novint Falcons. As consequence their coordinate systems are inverse, and hence the appearance of that characteristic x-axis symmetry.

It can be seen as well that the behavior of hand 3 is the worst one for both Novint Falcon (this could be future case of study).

4.4 Summary

The experiments carried out have successfully tested the software and hardware used and developed in this project. When testing the final integrated system some unexpected behaviors were observed as well. These results are explained by the use of both, Leap Motion and Novint Falcon technologies, with a finality they were not theoretically intended.

Software

The whole developed program has been successfully checked out. The defined virtual surface format gives to the system a robust and general input way of communication with the system. Leap Motion SDK and Novint Falcon SDK has been integrated in the same Visual Studio project without any complication. Moreover, the developed control system which involves hardware initialization, user finger tracking, prediction of surface contact point, calculation of Novint Falcon position and necessary applied forces to render the virtual surface with regard to the predicted contact point, workspaces and coordinate systems maching, and programme finalization, has been accurately tested, being in this way demonstrated its right functionality.

Aside this, many improvements are still available. A short list of these ones and future work as well on this field is provided in Section 5.2.

Hardware

When it comes to the final hardware platform some unexpected behaviors have been observed. Concretely, two are the main 'troubles' found, one regarding to the Leap Motion Controller and the other one to the haptic device developed by the MINT lab.

According to the Leap Motion Controller it's been observed that when it is faced to the Novint Falcon haptic device and to the StimTac, since they rest within its field of view, it detects and therefore interprets some parts of the device as finger or tools, because of its form, when in fact they are not. This obviously corrupts the correct operation of the system because it is supposed to find only one hand and one finger within its interaction area. As a consequence, when this happens an unpredictable response of the device takes place, which ruins and impoverishes the haptic experience of the user.

With respect to the Novint Falcon haptic device, the observated behavior in fact is not unexpected but worsens as well the haptic experience of the user when interacting with the developed device. As it's been stressed out in previous sections, the prototype of haptic device designed by the MINT lab in the context of Tao Zeng PhD is in a high grade physically limitated. This means that obviously not any random surface can be rendered but just some simple one. It also implicates that, even when the software part of the systems properly works as it's been highlighted in the section above, the physical constrains of the device finally provocates a not totally precise response from its part. This is easily understood by an example: let's imagine we want to simulate a surface defined by a given position, elevation an orientation of the end-effector plate. As we have seen along the report, first of all is calculated the position, then the orientation and finally the elevation is adjusted. According to this it is very easy to find that once fixed the orientation of the plate, is not possible to set the necessary elevation to simulate the surface, because doing it would implicates to lose the previously achieved orientation of the plate, or a wrong position. Or directly let's imagine a big orientation, such us 60º, which the actual prototype can't achieve. All this directly is translated to a low level of performance of the haptic device.

Potential Solutions

According to the previous sections, several potential solutions for the detected conflicting cases are provided along following lines.

In the context of Leap Motion drawbacks the idea is simple; taking into account that Leap Motion technology can be considered as newly created the efforts of the company in terms of evolution, development and improvement work are priority. Nowdays, a new version 2.0 Beta of its software is already accessible to Leap Motion developers. Thus, it is presimibable that close in the future the Leap Motion Controller tracking behavior will be much more accurate. This will mean a more precise differentiation between hands, finger and tools within the Leap interaction area, than it actually is, point that could be enough to overcame the observed problems when facing the Leap Motion Controller with the haptic feedback device. Morever, and with regard to the lab work, there are things to try. Even though the way the Leap Motion Controller treat the data detected in its field of view is not been revealed by the company, we know that it is built from two monochromatic IR cameras and three infrared LEDs (Section 2.1), which allow us to , by research work on materials, build the new prototype of the device, using, when and where necessary, some material that are not seen IR technology, becaming in this way invisible to the Leap Motion Controller and therefore overcoming the observed drawbacks.

Regarding to the haptic device troubles, and more precisely regarding to the problem of its very limitated range of movement (ROM), intuitively can be noticed that directly by the developing of the new prototype (stuff already being carried out) which will duplicate the degrees of freedom of the device from 3 to 6, plus the fact that will be built from newer technology, the ROM of this one will be bigger than the previous one, overcoming and this way many of the problems that appeared with the old prototype, if not all. Of course new restriction will appear, but those won't be the same one than the observed along this project, so that will be work for the lab team which keeps on working on this project.

Chapter 5
Conclusions

5.1 Summary

This project has set out to create an integration between the Leap Motion Controller and the Novint Falcon haptic device design by MINT lab where the Leap Motion technology allows to improve the general characteristics and behavior of this early prototype, more precisely, it acts as control platform of the haptic interaction device so that its end-effector pad, which is a tactile plate also developed by the Mint lab, can be continuously adapted to the user interaction and with regard to a virtual surface previous defined, to provide real shape and texture touch feeling to the user.

According to this, a successful implementation has been developed, by using the Leap Motion and the Novint Falcon SDK, resulting in a real time application which satisfies the main goal of the project. A general input format for the virtual surfaces to be rendered has been defined. The Leap Motion controller interaction area has been programmed to emulate that virtual surface, from this, a platform to predict the potential contact point of the user finger with that surface has been developed, and this data has been properly treated so that the tactile pad, which allows texture rendering, is in real time adapted to the user interaction, so that a real touch sensation is experienced by the user when interacting with the device.

During testing of the system, the source developed code presents real and reliable results. The behavior of the hardware platform correctly fulfills the main goal of the project as well, yet some unexpected events have been observed. Thus, a short list of feasible solutions has been provided (Section 4.5), which, given some time and adequate funding, can be done. A suggestion of possible improvements and future work on the developed prototype has been provided as well in the next section.

5.2 Future Work

As it's been highlighted several times along this report, the totality of the developed work of this project has been carried out with an early prototype of haptic device developed by MINT lab some years ago. However, another project which aims to build a new 6-DOF prototype of a similar device with the same and extra functionalities is already taking place in the context of the IRCICA MINT and Stimtac research group. Thus, all the suggestion of future work presented below and regarding to this project has been done within the framework of this new 6-DOF prototype since obviously, once this is done, the old one will become obsolete.

Plus, taking into account that this project was just a 'first approach' to what the global MINT project means, it must be noticed that the whole developed work in

terms of processing and treating data, virtual simulation or Leap Motion-Novint Falcon integration is subject to changes and improvements.

Software

In terms of software some of the possible work that could be developed in a near future is:

- Developent of a little GUI, by using the existing Leap Motion Software, so that the user can follow in a screen, in real time, his own hand and finger with regard to the virtual surface that is being rendering by the haptic device in order to make his experience when interacting with the device more intuitive and easy.

- Considering the rendering of big-size virtual surface, that is, surfaces with size is bigger that the ROM of the haptic device. This is somehow possible by implement a zoom and unzoom functionality with Leap Motion software (this is something already done in other applications such as Google Earth).

- Haptic force feedback could be used to give the user an additional level of interaction by implementing vibrations of different intensities when rendering a very detailed part of the surface cannot be performed.

- Rendering mixed concave-convex surfaces, which is something not done before.

- Considering the continuous adaptation of the end-effector pad from a point of view where not always the center of the plate has to match with the user finger. When the user moves his finger form one to another position where the final position is still within the surface of the tactile pad it is not necessary to modify the position of the plate and just the elevation and orientation. Approaching the problem in this way some computational time might be saved and with this a faster and real-time application would be implemented.

- Add some audio feedback for example when surface limits are reached or when touching user-pad takes place in order to enhance the user experience.

Hardware

In this field, as it’s been already commented, the development of a new 6-DOF prototype is already taking place. Within this framework the main necessity resides on integrate the Leap Motion Controller in that new version of the haptic device, task which, following the work presented along this project, must be easily achieved.

Appendix A
Leap Motion

Technical Specifications Summary

- Height: 0.5" (12.7 mm)
- Width: 1.2" (~30 mm)
- Depth: 3" (~80 mm)
- Weight: 0.1 pounds (~45 g)
- Included Cables: 24" and 60" USB 2.0 (microUSB 3.0 connectors)

Minimum System Requirements

- OS: Windows 7 or 8 or MAC OS X 10.7 Lion
- Processor: AMD PhenomTM II or Intel® CoreTM i3, i5, i7
- Graphic card: 256Mb 3D hardware accelerated graphics card
- Memory: 2 GB RAM
- USB 2.0 port
- Internet Connection

Warranty Terms

- 1 year limited

Included in the Package

- Leap Motion Controller
- 2 custom-length USB 2.0 cables
- Welcome card
- Important Information Guide

Appendix B
Novint Falcon

Technical Specifications Summary

- 3D Touch Workspace: 4" x 4" x 4" (4" = 10.2 cm)
- Force Capabilities: > 2 lbs (0.91 kg)
- Position Resolution: > 400 dpi
- Communication Interface: USB 2.0
- Size: 9" x 9" x 9" (9" = 22.86 cm)
- Weight: 6 lbs (2.7 kg)
- Power: 30 watts, 100 V - 240 V, 50 Hz - 60 Hz

Recommended System Requirements

- Processor: 2.4 GHz Processor
- OS: Windows XP Service Pack 2, Windows Vista
- Graphic card: 256Mb 3D hardware accelerated graphics card
- DirectX Version: DirectX 9.0c
- Hard Drive: 1.5 GB free disk space
- Memory: 1 GB RAM
- Broadband Internet Connection

Appendix C
Leap Motion Airspace Home

To discover and enjoy all application provided by Leap Motion Airspace follow the next steps:

On Windows

Airspace Home can be launched from the Leap Motion icon menu on the Windows Task bar: right-click on the icon and select "Launch Airspace". If the icon isn't visible, click the Show Hidden Icons button to see if Windows is hiding it. If it is still not visible, start your Leap Motion software.

If you checked "Create desktop shortcut" when you installed, you will have an Airspace icon on your Desktop.

During install, there is also an option to create a Start Menu folder; by default this is called "Leap Motion". This will create a folder with a shortcut in Start Menu → All Programs → Leap Motion.

You can also find Airspace Home installed in: C:\Program Files (x86)\Leap Motion\Airspace.

On Mac

Airspace Home can be launched from the Leap Motion icon menu on the Mac menu bar: click on Leap Motion icon and select "Launch Airspace."

Airspace Home will be in your Applications folder and it will be called "Airspace".

You can also search for the "Airspace" application using Spotlight (small magnifying glass in the top right corner of the screen).

Appendix D
General Leap Motion Project Setup

This appendix discusses how to set up and compile C++ projects from the command line and popular IDEs that can be found in the market, focusing on Windows OS which is the used along this project. Specifications and details for other OS such us Mac or Linux can be found in the official Leap Motion website [SET].

Compilers and libraries

On Windows, you can use the Visual C++ compiler included with Visual Studio 2008, 2010, or 2012. On OS X and Linux, you can use the gcc or clang compiler.

On Windows, the Leap Motion C++ API is provided in the dynamically linked library, Leap.dll (release) or Leapd.dll (debug). Separate libraries are provided for 32-bit, x86 architectures and 64-bit, x64 architectures.

The Leap Motion dynamic libraries are designed to be loaded from the same directory as your application executable. You are expected to distribute the appropriate Leap Motion library with your application. The Leap Motion libraries are located in the lib folder of the Leap SDK package.

Setting up a C++ project in Visual Studio

The Leap Motion SDK package includes sample projects for Visual Studio 2008, 20010, and 2012. You can use these projects as a starting point for your Leap Motion projects. This section illustrates how to create a project from scratch. Most of the steps also apply to adding Leap Motion support to an existing project. The example uses Visual Studio 2012.

To add Leap Motion support to a new or existing project:

Important: If you are creating a 64-bit application, use the libraries in the lib\x64 directory of the SDK, not the 32-bit libraries in the lib\x86 directory.

1. Open or create a new project of the desired type.

2. Make a LEAP_SDK system environment variable to point to your Leap Motion SDK folder. This step is optional, but does simplify creating references to the SDK files in your projects. (As a reminder, you can create and change system environment variables from the Windows System Properties dialog.)

3. Open the Project Property Pages using the Project > Properties menu command.

4. Set up a Debug configuration:

    a. Choose the Debug configuration.

    b. Add the SDK include directory to your project:

        i. Under Configuration Properties, select C/C++ > General.
        ii. In the properties panel, add the SDK include directory to the *Additional Include Directories* field by adding: $(LEAP_SDK)\include (where LEAP_SDK is the name of the environment variable you created earlier. You can substitute the appropriate file path if you do not want to use an environment variable).

    c. Add references to the Leap Motion libraries:

        i. Under Configuration Properties, select Linker > General.
        ii. In the properties pane, add the SDK lib\x86 directory (or lib\x64 for 64-bit configurations) to the *Additional Library Directories* field by adding: $(LEAP_SDK)\lib\x86
        iii. Select Linker > Input
        iv. Add Leapd.lib to the *Additional Dependencies* field.

    d. Add a *Post-Build Event* to copy the Leap Motion libraries to the project's target executable directory.

        i. Under Configuration Properties, select Build Events > Post-Build Event.
        ii. Edit the *Command Line* field to copy the libraries, adding:

            Xcopy /yr "$(LEAP_SDK)\lib\x86\Leapd.dll" $(TargetDir)"

            Xcopy /yr "$(LEAP_SDK)\lib\x86\msvcp100d.dll" $(TargetDir)"

            Xcopy /yr "$(LEAP_SDK)\lib\x86\msvcr100d.dll" $(TargetDir)"

            (The msvcp100d.dll and msvcr100d.dll libraries are only needed for debugging configurations. For release configurations, copy only Leap.dll.)

5. Set up a Release configuration:

    a. Choose the Release configuration.

    b. Add the SDK include directory as shown above.

    c. Add references to the Leap Motion libraries as shown above, except using Leap.lib (not Leapd.lib).

    d. Add a *Post-Build Event* to copy Leap.dll to the target executable directory.

6. Add your source code...

Appendix E
General Novint Falcon Software Setup

To begin using the Novint Falcon, the included tutorial and bundled games, it is recommended that you check the Novint website for the latest software and drivers before following the instructions below.

- Novint Website [NOV].
- Novint Community [FALB_FAL].
- Download and install Novint Falcon driver (v4.0.28) [DRV].
- Download and install F-Gen V1 [FGN].
- Ensure that NDSSetter (Novint Device Support Setter) is set as:

C:/Program Files/Novint/Falcon/HDAL (Figure C.1).

*** NOTE *** Novint Falcon drivers MUST be installed BEFORE F-Gen.

Once completed, you will be able to run the Novint FALCON DEMO (highly recommended) and play the included "Limited Edition Novint Falcon Bundle" through the NVeNT interface.

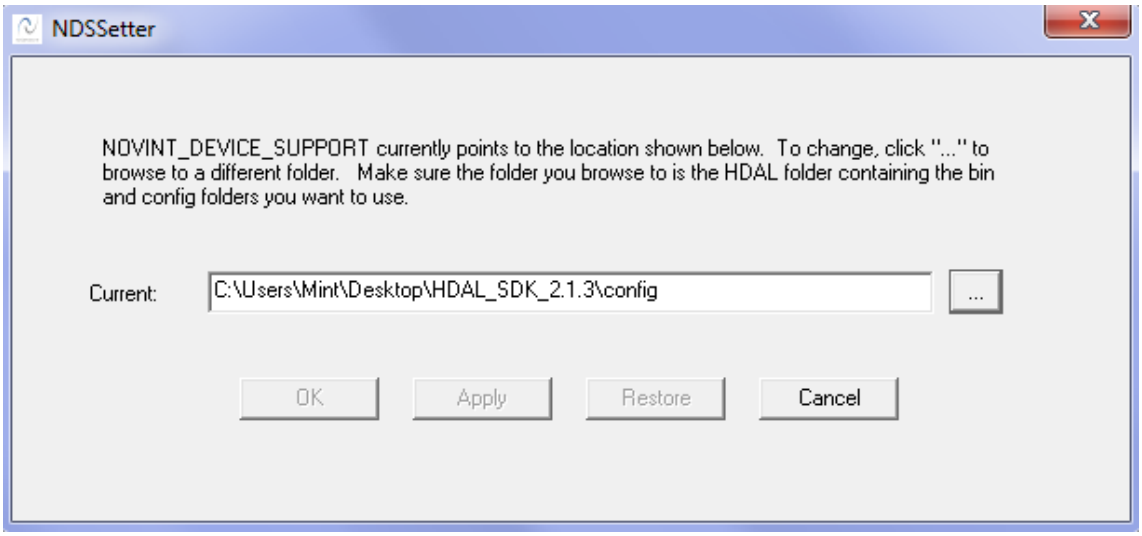


Figure C.1: NDSSetter Setup

Appendix F
General Squirrel Software Setup

Squirrel scripting is an excellent way for any beginner to become familiar with the Novint Falcon and its haptic capabilities. Novint uses Squirrel as the base programming language for its F-Gen interface which allows gamers to customize their Novint Falcon according to which game they are playing and what level of haptic feedback they wish to experience. Many functions are available to the user in addition to a large amount of control to how haptic forces are generated. The main disadvantage with Squirrel is that many functions that are available in C++ cannot be realized when purely using Squirrel.

With the Novint Falcon drivers and F-Gen already installed there are only two components needed in order to begin experimenting with the Novint Falcon and its haptic capabilities; Squirrel and an IDE. The IDE used along this project is Visual Studio version 2005.

- Download the latest version of Squirrel [SQR] and install.
- It is recommended you read Part 1, 2, 3, 4, 10 and 12 of the Basic Novint Falcon Tutorial using Squirrel Scripting to get started.

Appendix G
Novint Falcon SDK Setup

Follow these steps to properly install the Novint Falcon SDK (HDAL) on your computer:

- Ensure you have installed the drivers as discussed in section E.
- Connect the power and USB cables.
- Download the Novint Falcon SDK (HDAL) and install [SDKN].
- Set the path for the NDSSetter (Novint Device Support Setter) as: C:/Program Files/Novint/HDAL_SDK_2.1.3/config.

Global Search Setup

This procedure tells VS 2005 where to find additional directories which have needed include and library files. These settings apply to ALL development projects.

- Select "Tools" then "Options".
- In the new window, on the left hand side, select "Projects and Solutions", "VC++ Directories".
- On the right hand side under "Show Directories For" choose "Include Files" from the drop down menu.
- Double click on an empty slot then use the "...." icon to browse to the "C:/Program Files/Novint/HDAL_SDK_2.1.3/include" directory.
- In the same window, select "Library Files" in the drop down menu and repeat the process, this time navigating to the "C:/Program Files/Novint/HDAL_SDK_2.1.3/lib" folder.
- Click "OK".

Project Search Setup

This procedure tells VS 2005 where to find the necessary files in the Novint Falcon SDK which must be setup for each project using the Novint Falcon SDK (HDAL).

- Select "Project" then "Properties".

- In the new window, on the left hand side, select "Configuration properties", "C/C++", "General".
- On the right hand side, choose "Additional Include Directories" and add "$(NOVINT_DEVICE_SUPPORT)/include".
- In the same window on the left hand side, select "Configuration properties", "Linker", "General".
- On the right hand side, choose "Additional Library Directories" and add "$(NOVINT_DEVICE_SUPPORT)/lib".
- Click "OK".

Library File

- Navigate to "Project", "Properties", "Configuration Properties", "Linker", "Input" and add the "hdl.lib" file (copy it from the SDK location to your project "/lib" folder) to the "Additional Dependencies" window.

Appendix H

OpenGLUT

At this point, it is possible to run the included SDK "Basic" example which is described in the next section, however, a successful interaction with the Novint Falcon requires some form of visual representation and to acheive this, OpenGLUT was used to draw the graphical interface for this project. Installing OpenGLUT is a relatively simple process.

- Download and unpack OpenGLUT [OGL].
- Copy the *.h files to the VS 2005 compiler OpenGL "include" directory "C:/Program Files/Microsoft Visual Studio 8/VC/include/GL".
    1. openglut.h
    2. openglut_exp.h
    3. openglut_ext.h
    4. openglut_std.h
- Copy the *.lib files to the VS 2005 compiler "library" directory "C:/Program Files/Microsoft Visual Studio 8/VC/lib".
    1. OpenGLUT.lib
    2. OpenGLUT_static.lib
- The #include used in the code is — #include <GL/openglut.h>

Appendix I
Source Code

```
/*****************************************************************************\
*                                                                             *
*                              RUBEN LAENCINA ESCOBAR                         *
*                               Final Degree Project                          *
*                            IRCICA, MINT - StimTac lab                       *
*                                 April-July, 2014                            *
*                                                                             *
\*****************************************************************************/

#include <iostream>
#include <math.h>
#include <fstream>
#include <windows.h>

using namespace std;

/*****************************************************************************\
*                                  Novint Falcon                              *
\*****************************************************************************/

#include <hdl/hdl.h>
#include <hdlu/hdlu.h>

// Serial Number of Novint Falcon Devices
// SN1 = 14100263
// SN2 = 14QAXQ2X

// Global variables
#define XROM 0.10    // Range of Movement (ROM) in x-axis in metres
#define YROM 0.10    // Range of Movement (ROM) in y-axis in metres
#define ZROM 0.04    // Range of Movement (ROM) in z-axis in metres
#define ZMIN -0.06   // Min Z
#define ZMAX -0.02   // Max Z = ZMIN + ZROM
#define F_LIM 30.0   // Range of force

// Handle to device
HDLDeviceHandle deviceHandle;
HDLDeviceHandle deviceHandle2;

// Handle to Contact Callback
HDLServoOpExitCode servoOp;

// Variables used by servo thread
bool   button, button2;
const bool bNonBlocking = false;

// Variable used to Control Falcons' Status
static double position[3], position2[3];
static double force[3] = {0.0,0.0,0.0}, force2[3] = {0.0,0.0,0.0};
static double pos_previous[3], pos2_previous[3];
static double velocity[3] = {0.0,0.0,0.0}, velocity2[3] = {0.0,0.0,0.0};
```

```
static double velocity_previous[3] = {0.0,0.0,0.0}, velocity2_previous[3] =
{0.0,0.0,0.0};

static double x_ref = 0.0, y_ref = 0.0, z_ref = -0.04, x2_ref = 0.0, y2_ref =
0.0, z2_ref = -0.04;// Initial Falcons' Position
static double e[3]={0.0,0.0,0.0}, e2[3]={0.0,0.0,0.0};
static double deta_I[3]={0.0,0.0,0.0}, deta2_I[3]={0.0,0.0,0.0};
static double I[3]={0.0,0.0,0.0}, I2[3]={0.0,0.0,0.0};
static double I_previous[3]={0.0,0.0,0.0}, I2_previous[3]={0.0,0.0,0.0};
static double P[3]={0.0,0.0,0.0}, P2[3]={0.0,0.0,0.0};
static double V[3]={0.0,0.0,0.0}, V2[3]={0.0,0.0,0.0};

float novintCoordenates[3];
float orientationRad = 0;  // In Radians
float orientationDeg = 0;  // In Degrees

static const float d = 0.01;
static const float T = 0.001;
int k1=100, k2=1500, k3=50;

// Output files
ofstream falcon1, falcon2, realfalcon1, realfalcon2, shape, measurements;
int count = 0, count2 = 0;

// Set position of falcon1
void setFalcon1(double pos[3], double force[3])
{
       //  Tustin Trqnsformation: v=[2/(T+2*d)]*[X(n)-X(n-1)]-[(T-
2*d)/(T+2*d)]*V(n-1)

       velocity[0] = (2/(T+2*d))*(pos[0]-pos_previous[0])-((T-
2*d)/(T+2*d))*velocity_previous[0];
       velocity[1] = (2/(T+2*d))*(pos[1]-pos_previous[1])-((T-
2*d)/(T+2*d))*velocity_previous[1];
       velocity[2] = (2/(T+2*d))*(pos[2]-pos_previous[2])-((T-
2*d)/(T+2*d))*velocity_previous[2];

       e[0] = x_ref - pos[0];
    e[1] = y_ref - pos[1];
    e[2] = z_ref - pos[2];

       deta_I[0] = k1*T*e[0];
       deta_I[1] = k1*T*e[1];
       deta_I[2] = k1*T*e[2];

       I[0] = I_previous[0] + deta_I[0];
       I[1] = I_previous[1] + deta_I[1];
       I[2] = I_previous[2] + deta_I[2];

       P[0] = k2*e[0];
       P[1] = k2*e[1];
       P[2] = k2*e[2];

       V[0] = k3*(velocity[0]);
       V[1] = k3*(velocity[1]);
       V[2] = k3*(velocity[2]);

    force[0] = P[0] - V[0] + I[0];
    force[1] = P[1] - V[1] + I[1];
    force[2] = P[2] - V[2] + I[2];

       // Forces Limitation
```

```
    if (force[0] > F_LIM) force[0] = F_LIM;  if (force[0] < -F_LIM) force[0] = -
F_LIM;
    if (force[1] > F_LIM) force[1] = F_LIM;  if (force[1] < -F_LIM) force[1] = -
F_LIM;
    if (force[2] > F_LIM) force[2] = F_LIM;  if (force[2] < -F_LIM) force[2] = -
F_LIM;

    // Previous Variables Update
       I_previous[0] = I[0];
       I_previous[1] = I[1];
       I_previous[2] = I[2];

    pos_previous[0] = pos[0];
       pos_previous[1] = pos[1];
       pos_previous[2] = pos[2];

    velocity_previous[0] = velocity[0];
       velocity_previous[1] = velocity[1];
       velocity_previous[2] = velocity[2];
}

// Set position of falcon2
void setFalcon2(double pos[3], double force[3])
{
       //  Tustin Trqnsformation: v=[2/(T+2*d)]*[X(n)-X(n-1)]-[(T-
2*d)/(T+2*d)]*V(n-1)

       velocity2[0] = (2/(T+2*d))*(pos[0]-pos2_previous[0])-((T-
2*d)/(T+2*d))*velocity2_previous[0];
       velocity2[1] = (2/(T+2*d))*(pos[1]-pos2_previous[1])-((T-
2*d)/(T+2*d))*velocity2_previous[1];
       velocity2[2] = (2/(T+2*d))*(pos[2]-pos2_previous[2])-((T-
2*d)/(T+2*d))*velocity2_previous[2];

       e2[0] = x2_ref - pos[0];
    e2[1] = y2_ref - pos[1];
    e2[2] = z2_ref - pos[2];

       deta2_I[0] = k1*T*e2[0];
       deta2_I[1] = k1*T*e2[1];
       deta2_I[2] = k1*T*e2[2];

       I2[0] = I2_previous[0] + deta2_I[0];
       I2[1] = I2_previous[1] + deta2_I[1];
       I2[2] = I2_previous[2] + deta2_I[2];

       P2[0] = k2*e2[0];
       P2[1] = k2*e2[1];
       P2[2] = k2*e2[2];

       V2[0] = k3*(velocity2[0]);
       V2[1] = k3*(velocity2[1]);
       V2[2] = k3*(velocity2[2]);

       force[0] = P2[0] - V2[0] + I2[0];
       force[1] = P2[1] - V2[1] + I2[1];
       force[2] = P2[2] - V2[2] + I2[2];

       // Forces Limitation
       if (force[0] > F_LIM) force[0] = F_LIM;  if (force[0] < -F_LIM) force[0] =
-F_LIM;
```

```
	if (force[1] > F_LIM) force[1] = F_LIM;  if (force[1] < -F_LIM) force[1] =
-F_LIM;
	if (force[2] > F_LIM) force[2] = F_LIM;  if (force[2] < -F_LIM) force[2] =
-F_LIM;

	// Previous Variables Update
	I2_previous[0] = I2[0];
	I2_previous[1] = I2[1];
	I2_previous[2] = I2[2];

	pos2_previous[0] = pos[0];
	pos2_previous[1] = pos[1];
	pos2_previous[2] = pos[2];

    velocity2_previous[0] = velocity2[0];
	velocity2_previous[1] = velocity2[1];
	velocity2_previous[2] = velocity2[2];
}

// Test error function
void testHDLError()
{
	HDLError err = hdlGetError();
	if (err != HDL_NO_ERROR)
	{
		std::cout << "HDLError " << err << std::endl;
		abort();
	}
}

// Non blocking callback function - servo loop
HDLServoOpExitCode NonBlockingServoOpCallback(void* pUserData)
{
	// Novint Falcon 2

	hdlMakeCurrent(deviceHandle);
	hdlToolButton(&(button));
    hdlToolPosition(position);
	setFalcon1(position, force);
	hdlSetToolForce(force);

     testHDLError();

	// Novint Falcon 2

	hdlMakeCurrent(deviceHandle2);
	hdlToolButton(&(button2));
    hdlToolPosition(position2);
	setFalcon2(position2, force2);
	hdlSetToolForce(force2);

    testHDLError();

	// Enable Novint Falcons' Button to quit
	if (button || button2)
	{
		cout << "\n" << "Button pressed on any haptic device." << endl;
		exit(1);
	}

	// Make sure to continue processing
	return HDL_SERVOOP_CONTINUE;
```

```
}

// Novint Falcon Setup
int setNovintFalcon()
{
       // Number of haptic devices currently connected to the computer
    int numHapticDevices = hdlCountDevices();

    // Exit if less than necessary haptic devices found
       if (numHapticDevices < 2)
       {
              cout << "Connected haptic devices: " << numHapticDevices << ". Not
enough connected haptic devices. Exit." << endl;
              Sleep(1500);
              exit(1);
       }

    // Inits haptic devices
    deviceHandle = hdlInitNamedDevice("FALCON_1");
       deviceHandle2 = hdlInitNamedDevice("FALCON_2");
    testHDLError();

    if (deviceHandle == HDL_INVALID_HANDLE || deviceHandle2 ==
HDL_INVALID_HANDLE)
    {
        std::cout << "Could not open device: HDL_INVALID_HANDLE" << std::endl;
        exit(1);
    }

    // Starts servo and all haptic devices
    hdlStart();

    // Sets callback for the nonblocking servo loop
    servoOp = hdlCreateServoOp(NonBlockingServoOpCallback, NULL, bNonBlocking);
       testHDLError();

    if (servoOp == HDL_INVALID_HANDLE)
   {
        std::cout << "Invalid servo op handle: HDL_INVALID_HANDLE" << std::endl;
        exit(1);
   }

       return 0;
}

/*****************************************************************************\
*                                    Leap Motion                              *
\*****************************************************************************/

#include "Leap.h"
using namespace Leap;

// Global variables
#define XLIMIT 180  // X-axis Leap Motion limit
#define ZLIMIT 180  // Z-axis Leap Motion limit
#define RESOLUTION 1       // Resolution used to predict the point of contact
#define PIXSIZE 2*XLIMIT/RESOLUTION

// Define the virtual surface to be simulated
class Surface
{
       int dim_x;
```

```
        int dim_z;
        float **surface;
        float pictureResolution;
        int center[2];
        float max;

public:

        virtual int getX() { return dim_x; }
        virtual int getZ() { return dim_z; }
        virtual float getMax() { return max; }
        virtual int* getCenter() { int *Center = center; return Center; }
        virtual float getPictureResolution() { return pictureResolution; }
        virtual float** defineSurface(int, int);
        virtual float** getSurface() { return surface; }
};

// Global Virtual Surface
Surface surf;

float** Surface::defineSurface(int x, int z)
{
        dim_x = x;
        dim_z = z;
        center[0] = x / 2;
        center[1] = z / 2;

        // 2D Matrix that will contain the surface data
        float **picture = new float*[x];
        for (int i = 0; i < x; i++)picture[i] = new float[z];

        // Pixel's size of the picture
        pictureResolution = float(x) / float(PIXSIZE);

        float z_pic = 0;
        float x_pic = -(x-1)/2;
        max = 0;

        for (int i = 0; i < x; i++)
        {
                for (int j = 0; j < z; j++)
                {
                        float aux = 10 - pow(x_pic,2)/250;       // Mathematical
expression of the shape to be simulated

                        if (aux < 0) picture[j][i] = 0;
                        else { picture[j][i] = aux; }

                        x_pic += 1;

                        if (picture[j][i] > max) max = picture[j][i];   // Calcula
the heightest value of the picture
                }
                x_pic = - (x-1)/2;
        }

        surface = picture;
        return surface;
}

// Leap Motion and Novint Falcon workspace matching
void leapMotionToNovintFalcon(int xContactPix, int zContactPix)
```

```
{
	// Transformation from Leap Motion to Novint Falcon coordinate system
	float x = XROM/surf.getX();
	float z = YROM/surf.getZ();

	float xCenter = surf.getCenter()[0] * x;
	float zCenter = surf.getCenter()[1] * z;

	x =  (x * xContactPix - xCenter);
	z = z * zContactPix - zCenter;
	float y = ZROM/surf.getMax() * surf.getSurface()[xContactPix][zContactPix]
+ ZMIN;

	novintCoordenates[0] = x;
	novintCoordenates[1] = z;
	novintCoordenates[2] = y;

	// Calculation of final Stimtac position, elevation and orientation

	x_ref = novintCoordenates[1];
	y_ref = -novintCoordenates[0];
	z_ref = novintCoordenates[2];

	x2_ref = -x_ref;
	y2_ref = -y_ref;
	z2_ref = novintCoordenates[2];

	// Condition to control initial and last pixel
	                                                                        //
cuidado con esto para el menor
	int xContactPixPost = xContactPix + 1;
	if(xContactPixPost > surf.getX()-1) xContactPixPost = surf.getX()-1;

	// Get height difference between the contact pixel and its next one
	float prev = surf.getSurface()[xContactPix][zContactPix];
	float next = surf.getSurface()[xContactPixPost][zContactPix];
	float difZ = next - prev;

	orientationRad = atan(abs(difZ)/surf.getPictureResolution());	// In
Radians
	orientationDeg = orientationRad * 360 / (2*PI); // In Degrees

	float plate = 0.076;
	float H = sin(orientationRad)*plate + ZMIN;

	// Differencite the sense of the orientation
	float aux = 0;
	float aux2 = 0;

	if(difZ > 0)
	{
		aux2 = H;
		aux = ZMIN;
	}
	else {
		aux2 = ZMIN;
		aux = H;
	}

	// Compensation of elevation
	float compensation = ((H + ZMIN)/2) - novintCoordenates[2];
```

```
        aux -= compensation;
        aux2 -= compensation;

        // Novint Falcon safety limits
        if(aux > ZMAX) aux = ZMAX;
        if(aux < ZMIN) aux = ZMIN;
        if(aux2 > ZMAX) aux2 = ZMAX;
        if(aux2 < ZMIN) aux2= ZMIN;

        // Final assignation of the elevation of both Novint Falcon
        z_ref = aux;
        z2_ref = aux2;
}

// Leap Controler Listener implementation
class SampleListener : public Listener {
public:
        virtual void onInit(const Controller&);
        virtual void onConnect(const Controller&);
        virtual void onDisconnect(const Controller&);
        virtual void onExit(const Controller&);
        virtual void onFrame(const Controller&);
        virtual void onFocusGained(const Controller&);
        virtual void onFocusLost(const Controller&);
};

void SampleListener::onInit(const Controller& controller) {
        // Printed by screen when System Successfully Initialized
        cout << "\n" << "System Successfully Initialized. " << "\n" << endl;
}

void SampleListener::onConnect(const Controller& controller) {
        cout << "Leap Connected" << endl;
}

void SampleListener::onDisconnect(const Controller& controller) {
        cout << "Leap Disconnected" << endl;
}

void SampleListener::onExit(const Controller& controller) {
        cout << "System Successfully Exited." << endl;
}

void SampleListener::onFrame(const Controller& controller)
{
        // Get the most recent frame
        const Frame frame = controller.frame();
        if (!frame.hands().isEmpty()) {
                // Get the first hand
                const Hand hand = frame.hands()[0];

                // Check if the hand has any fingers
                const FingerList fingers = hand.fingers();
                if (!fingers.isEmpty()) {

                        // Take the first finger
                        Finger finger = fingers[0];

                        // Stimate point of contact
                         int xContact = 0;
                        int zContact = 0;
```

```
                    if (fingers[0].tipPosition()[0] > -XLIMIT &&
fingers[0].tipPosition()[0] < XLIMIT){
                        xContact = floor(fingers[0].tipPosition()[0] /
RESOLUTION)*RESOLUTION + RESOLUTION / 2;
                    }
                    if (fingers[0].tipPosition()[2] > -ZLIMIT &&
fingers[0].tipPosition()[2] < ZLIMIT){
                        zContact = floor(fingers[0].tipPosition()[2] /
RESOLUTION)*RESOLUTION + RESOLUTION / 2;
                    }

                    // Not expected contact if user finger is out of ROM
                    if (xContact == 0 || zContact == 0) cout << "Not expected
contact." << endl;
                    else {

                        cout << "Finger approaching surface at: (x = "  <<
fingers[0].tipPosition()[0] << ", z = "
                        << fingers[0].tipPosition()[2] << ")." << endl;

                        cout << "Potential contact point: (x = " << xContact
<< ", z = "   << zContact << ")." << endl;

                        // Stimate pixel of contact
                        int xContactPix = (xContact /
RESOLUTION)*surf.getPictureResolution() + surf.getPictureResolution() / 2 +
surf.getCenter()[0];
                        int zContactPix = (zContact /
RESOLUTION)*surf.getPictureResolution() + surf.getPictureResolution() / 2 +
surf.getCenter()[1];

                        leapMotionToNovintFalcon(xContactPix, zContactPix);

                        cout << "Potential contact pixel: (" << xContactPix <<
","    << zContactPix << ")." << endl;
                        cout << "Surface Elevation: y = " <<
surf.getSurface()[xContactPix][zContactPix] << "." << endl;
                        cout << "Surface Orientation (degrees): " <<
orientationDeg << ".\n" << endl;
                    }
            }
        }
}

void SampleListener::onFocusGained(const Controller& controller) {
        cout << "Focus Gained\n" << endl;
}

void SampleListener::onFocusLost(const Controller& controller) {
        cout << "\nFocus Lost" << endl;
}

//********************************** main ************************************\

void main()
{
        // Define Virtual Surface
        cout << "Defining virtual surface." << "\n" << endl;
        surf.defineSurface(101, 101);

        // Haptic Setup
        setNovintFalcon();
```

```
        Sleep(1500);

        // Leap Motion Setup
        SampleListener listener;
        Controller controller;
        controller.addListener(listener);

        // Keep this process running until Enter or any button on the haptic
devices is pressed
        cout << "Press 'Enter' or any button on the haptic devices to quit..." <<
"\n" << endl;
        Sleep(1500);
        cin.get();

        // Exit Process
        controller.removeListener(listener);
}
```

Bibliography

[1] F. Sato, H. Kajimoto, N. Kawakami, and S. Tachil, "Electrotactile display for integration with kinesthetic display", In Proceedings of 16th IEEE International Conference on Robot and Human Interactive Communication, Aug. 2007, pp. 3–8.

[2] E. P. Scilingo, M. Bianchi, G. Grioli, and A. Bicchi, "Rendering softness: Integration of kinesthetic and cutaneous information in a haptic device", IEEE Transactions on Haptics, vol. 3, no. 2, pp. 109–118, 2010.

[3] T. Zeng, B. Lemaire-Semail, F. Giraud, M. Messaoudi, and A. Bouscayrol, "Position control of a 3 DOF platform for haptic shape rendering", Power Electronics and Motion Control Conference (EPE/PEMC), 2012 15th International, vol., no., pp.LS6c.2-1, LS6c.2-5, 4-6 Sept. 2012.

[4] T. Zeng, "Conception et Contrôle d'un périphérique dédié à la simulation couplée kinesthésique et tactile", PhD Thesis, Jan. 2012.

[5] M. Biet, F. Giraud, and B. Lemaire-Semail, "Squeeze film effect for the design of an ultrasonic tactile plate", IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control, vol. 54, no. 12, pp. 2678–2688, Dec. 2007.

[6] M. Biet, F. Giraud, B. Lemaire-Semail, "Implementation of tactile feedback by modifying the perceived friction", The European Physical Journal - Applied Physics, Vol. 43, N°. 1, pages. 123-135, 7-2008.

[7] T. Zeng, B. Lemaire-Semail, F. Giraud, Member, IEEE, and M. Amberg, "Contribution of Slip Cue to Curvature Perception through Active and Dynamic Touch", IEEE Transactions on Haptics, Vol. 6, No. 4, October-DEecember 2013.

[8] M. Sinclair, M. Pahud and H. Benko, "TouchMover 2.0 - 3D Touchscreen with Haptic Feedback and Haptic Texture", Proc. of IEEE Haptics Symposium, 2014.

[9] Overholt, D., Pasztor, E., Mazalek, A., "Multipurpose Array of Tactile Rods for Interactive eXpression", in Conference Abstracts and Applications of SIGGRAPH '01 (Los Angeles, California, USA, August 12-17, 2001), ACM Press, p.232.

[10] M. Bordegoni, U. Cugini, M. Covarrubias, and M. Antolini, "A Force and Touch Sensitive Self-deformable Haptic Strip for Exploration and Deformation of Digital Surfaces", Lecture Notes in Computer Science Volume 6192, 2010, pp 65-72.

[11] Y. Matoba, T. Sato, N. Takahashi, and H. Koike, "ClaytricSurface: an interactive surface with dynamic softness control capability", In Proceedings of ACMSIGGRAPH2012, Emerging Technologies, 2012.

[12] M. Wiertlewski, J. Lozada, E. Pissaloux, and V. Hayward, "Causality Inversion in the Reproduction of Roughness, Haptics: Generating and Perceiving Tangible Sensations", Lecture Notes in Computer Science Springer Berlin Heidelberg, 2010, pages 17-24.

[13] G. Robles-De-La-Torre, "Comparing the role of lateral force during active and passive touch. Lateral force and its correlates are inherently ambiguous cues for shape perception under passive touch conditions", Eurohaptics Conference, pages 159-164,2002.

[14] Campion, G. and Hayward, "Fast calibration of haptic texture algorithms", IEEE Transactions on Haptics, Vol. 2, No. 2, Pages 85-93, 2009.

[15] G. Casiez, N. Roussel, R. Vanbelleghem and F. Giraud. Surfpad: riding towards targets on a squeeze film effect. In Proceedings of CHI'11, p. 2491-2500, May 2011, ACM.

[16] B. G. Becker, and L. Nelson, "Smooth transitions between bump rendering algorithms", In Proc. of the 20th annual conference on Computer graphics and interactive techniques (SIGGRAPH '93). ACM, New York, NY, USA, 183-190, 1993.

[AIR] https://airspace.leapmotion.com/

[BLG] https://www.leapmotion.com/blog/

[COM] https://community.leapmotion.com/

[CTL]

https://developer.leapmotion.com/documentation/cpp/supplements/Leap_Application.htm

[DEV] https://developer.leapmotion.com/

[DRV] https://backup.filesanywhere.com/fs/v.aspx?v=896a6886606576b0a368

[DVZ]

https://developer.leapmotion.com/documentation/cpp/supplements/Leap_Visualizer.html

[GLU1] http://www.openglut.sourceforge.net/index.html

[GLU2] http://www.opengl.org/resources/libraries/glut

[FAL] http://www.falconarmy.com

[FGN] https://backup.filesanywhere.com/fs/v.aspx?v=896a68865a6373afa59f

[LM] https://www.leapmotion.com/

[NOV] http://www.novint.com

[OGL] www.sourceforge.net/projects/openglut/files/

[PRD] https://www.leapmotion.com/product

[SDKL] https://developer.leapmotion.com/downloads

[SDKN] https://backup.filesanywhere.com/fs/v.aspx?v=896d658b5c6272b2a1aa

[SETL] https://www.leapmotion.com/setup

[SETPJ]

https://developer.leapmotion.com/documentation/cpp/devguide/Project_Setup.html

[SQR] http://www.squirrel-lang.org

[TUTL]

https://developer.leapmotion.com/documentation/cpp/devguide/Sample_Tutorial.html