Framework para la captura de eventos mediante interfaz natural de usuario

Juan Gabriel Fernandez[1], Gustavo Pereira[1], Jorge Ierache[1]

[1]Instituto de Sistemas Inteligentes y Enseñanza Experimental de la Robótica (ISIER)
Facultad de Informática Ciencias de la Comunicación y Técnicas Especiales
Universidad de Morón, Cabildo 134, (B1708JPD) Morón, Buenos Aires, Argentina
54 11 5627 2000 int. 189
jgfer@hotmail.com, pereira.gustavo@gmail.com, jierache@unimoron.edu.ar

Abstract: En el siguiente artículo se presenta una propuesta de framework abierto que permite integrar el uso de interfaces naturales para el control de objetos.
La creación de este framework surge a raíz de la problemática encontrada en las diferentes soluciones que existen y sus respectivos frameworks asociados; Estas soluciones permiten el empleo de interfaces naturales en diferentes entornos, sin embargo todas ellas son aplicaciones nativas encapsuladas para cumplir un único propósito. Por tal motivo surgió la necesidad de la creación de un framework abierto para el control de objetos y la presentación del concepto de monitores que sustenta la creación del mismo. En particular se desarrollaron casos de prueba para el control de robots.

Keywords: Kinect, Interfaz Natural de Usuario, Framework, Control, Robots.

1 Introducción

Una interfaz natural de usuario es aquella que se utiliza para interactuar con un sistema, aplicación o dispositivo sin utilizar un mando o control de entrada (por ejemplo: ratón, teclado, joystick, etc.) siendo reemplazados directamente por la captura de los movimientos del cuerpo humano, tales como las manos, las piernas, la cabeza, etc. En los últimos años, la popularidad de las interfaces naturales, ha experimentado un crecimiento abrumador. Estas interfaces permiten a un usuario interactuar con una computadora o dispositivo de forma sencilla y sin la necesidad de un aprendizaje complejo y/o especializado. Entre los principales usos que se le ha dado al sensor Kinect se encuentra el de controlar objetos o sistemas mediante gestos. Esto se logra generando aplicaciones dedicadas a tal el fin. Existen varios desarrollos realizados con el Kinect para el control de objetos o sistemas, como por ejemplo el control de mapas virtuales, operaciones aritméticas básicas en Windows o control a distancia de robot con servomotores. Los desarrollos anteriormente mencionados utilizan el sensor Kinect para dar solución a una problemática determinada y se limitan a un único uso. Esto genera la necesidad de la creación de nuevos desarrollos que permitan ampliar las funcionalidades.

En este trabajo se propone y presenta la creación de un framework para el uso genérico de los datos capturados por el sensor Kinect de Microsoft [1], sumando la capacidad de generar una base de monitores abierta para cada punto capturado con independencia del dominio de aplicación; un monitor es un estado que determina si un punto (llamado joint para la nomenclatura que utiliza Kinect) se encuentra dentro de un rango de valores permitido. Estos monitores de puntos capturados facilitan la declaración de métodos que activan eventos locales o remotos en función del estado del monitor, estos se categorizan en estáticos y dinámicos. Se muestra en el siguiente artículo el estado actual de los desarrollos y frameworks realizados para Kinect para el control de objetos, para luego analizar sus problemáticas y plantear una solución. Además se exhibe el desarrollo del framework abierto y sus casos experimentales. Finalmente se presenta la conclusión y las futuras líneas de investigación.

2 Situación actual

El sensor de movimiento o sensor 3D Kinect es un dispositivo de control por movimiento creado originalmente para poder interactuar con la consola de juegos Microsoft Xbox 360[2] sin necesidad de ningún mando o control. La cámara RGB permite generar una imagen tridimensional de lo que tiene por delante. Para ello cuenta con un sensor de profundidad, que permite ver una habitación en tres dimensiones. Este sensor emite rayos infrarrojos que se proyectan sobre la escena con un patrón de 50.000 puntos, invisibles al ojo humano, con el fin de marcar las líneas de profundidad, medir el tiempo que demora en retornar el haz de luz con respecto al dispositivo y calcular la distancia al objeto identificado. Kinect permite la captura de movimiento a través de una serie de algoritmos como son: el **Randomized Decision Forests [3],** que es capaz de aprender ciertas características de un conjunto de imágenes de profundidad con el fin de clasificar los pixeles de una imagen para obtener la pose de un objeto y el **Mean-Shift [4]** que es capaz de estimar la posición correspondiente de las articulaciones del esqueleto. Aplicando dichos algoritmos la tecnología con la que cuenta Kinect es capaz de realizar el reconocimiento esquelético de un individuo en tiempo real [5] [6]. De las diferentes alternativas existentes a Kinect, en el uso de interfaces naturales, se pueden destacar las siguientes: Perceptive Pixel [7], Microsoft PixelSense [8], 3D Immersive Touch [9] y MYO [10]; las cuales permiten captar los gestos y movimientos del cuerpo humano utilizando diferentes tecnologías y técnicas.

2.1 Aplicaciones desarrolladas para Kinect

a) *Touch screen con Kinect* desarrollado por la firma Ubi Interactive [11]; ésta, interactuando con Microsoft, desarrolló un sistema que utiliza un proyector digital combinado con el sensor Kinect para crear una pantalla táctil virtual, que puede ser proyectada en cualquier superficie. Mediante la captura de los movimientos de las manos, se permite el control de las imágenes proyectadas. Teniendo un proyector y el sensor de

movimiento Kinect solo resta una PC con la aplicación que permite convertir cualquier superficie en táctil.
b) Kinect para estudiar anatomía (Sistema Mirracle)[12]. Sistema creado en la Universidad Técnica de Münich, y permite instaurar la ilusión en la cual el usuario tiene "visión de rayos-x" en su propio cuerpo. Utilizando el Kinect de Microsoft para estimar la posición del usuario, y la base de datos del proyecto Visible Korean Human, la persona puede utilizar la interfaz de Kinect para navegar por su propio cuerpo. Existen trabajos de divulgación científica que presentan soluciones para el control de objetos o sistemas mediante la interface natural Kinect, tales como:
c) Wiimote and Kinect: Gestural User Interfaces add a Natural third dimension to HCI [13]. Las aplicaciones propuestas en este trabajo adoptan el seguimiento del movimiento del usuario a través de la Nintendo Wii y el sensor Kinect. Estas aplicaciones están específicamente diseñadas para la interacción gestual del usuario en mapas geográficos en 3D, como los utilizados por el motor de búsqueda Bing.
d) Robust Hand Gesture Recognition with Kinect Sensor[14]. Presenta el reconocimiento preciso y eficiente de gestos de la mano para aplicaciones básicas de la vida real, tales como: el cálculo aritmético y el juego piedra, papel o tijeras.
e) Human Machine Interface HMI using Kinect sensor to control a SCARA [15]. Plantea la posibilidad de manipular el robot SCARA (Selective Compliance Assembly Robot Arm) en tiempo real a través del sensor Kinect. Este desarrollo envía comandos por puerto USB a la controladora del SCARA.

3 Problemática

La problemática que se presenta en los desarrollos que existen, es la falta de generalidad a la hora de utilizar el sensor Kinect. Normalmente las aplicaciones creadas para Kinect determinan la forma en que se realizará la captura, estas definen los parámetros de ingreso y no pueden ser modificados por el usuario. De la misma forma lo hace con la salida o acciones que se realicen. Esto hace que estas aplicaciones tengan un uso bien definido y cerrado. Este acoplamiento no permite utilizar el sensor más que para su uso definido por la aplicación.

4 Solución Planteada

Se desarrolló un framework para uso genérico de los datos capturados por el sensor Kinect. El mismo, permite la creación de monitores para cada punto capturado por el sensor Kinect. Un monitor, como fue definido anteriormente, es un estado que determina si un punto (joint) se encuentra dentro de un rango de valores permitido. Para simplificar el concepto se usara uno de los joints entregados por el sensor Kinect. En un frame[1] determinado (de los treinta

[1] Se denomina frame a un fotograma o cuadro, una imagen particular dentro de una sucesión de imágenes que componen una animación. En esta caso la sucesión de datos que envía Kinect

por segundo que entrega el sensor Kinect) se tienen las coordenadas cartesianas (X, Y, Z) del joint seleccionado. Con esta información se puede hacer un análisis de las tres dimensiones del espacio, y a fin de simplificar el concepto solo se utilizarán las coordenadas X e Y. Con las capturas obtenidas se comienza la etapa de análisis de monitores. Se va a generar un monitor para el joint analizado. El monitor tendrá tres estados posibles: Activos, Inactivo, Nulo. Los estados Activo e Inactivo irán cambiando según se cumplan determinadas condiciones. El estado Nulo se refiere a la inhabilitación del monitor, lo que significa que si no se requiere su utilización los estados activo e inactivo no estarán disponibles. Para determinar que un monitor está activo o inactivo se definirá un rango de valores por cada eje cartesiano. De esta manera se tiene para X e Y un valor inicial (Xi), (Yi) y un valor final (Xf), (Yf). Teniendo definidos los rangos de X e Y se puede analizar el estado del monitor. El monitor relacionado con el punto seleccionado estará activo cuando los valores de X e Y estén comprendidos dentro de sus respectivos rangos: $Xi < X\ capturado < Xf;\ Yi < Y\ capturado < Yf$. Cuando el monitor esté dentro del rango de valores en un determinado frame, se lo considera como activo, en caso contrario su estado será el de inactivo. Se definió una región de valores en donde el monitor puede estar activo. A estos monitores los denominamos Estáticos. De esta manera se pueden generar N cantidad de monitores que estén relacionados a cada joint que entrega el Kinect. Cuando un monitor se active, se disparará la ejecución de uno o más métodos que serán generados para un fin determinado. Estos métodos desencadenarán acciones que podrán ser definidas para cumplir con un fin determinado. Estas acciones podrán ser ejecutadas dentro de un ámbito local o de manera remota. Se puede definir la cantidad de monitores y métodos que se deseen y relacionarlos con un punto en particular del cuerpo.

Continuando con el análisis de los monitores se presenta el caso donde se quiere conocer en qué momento una persona, que se encuentra delante del Sensor Kinect, está quieta, avanzando o retrocediendo. Con el concepto de monitor estático no es posible tener esta información ya que no basta con saber si un monitor se activa o no en un momento determinado. Se debe tener en cuenta cuál era la posición anterior para compararla con la posición actual y allí decidir qué acción se está realizando. Si se analiza la distancia entre el sensor y la persona se tiene que: Si el valor de la posición actual (Z actual) es menor que el valor de la posición anterior (Z anterior) entonces la persona está avanzando (Fig. 1). Si el valor de la posición actual (Z actual) es mayor que el valor de la posición anterior (Z anterior) entonces la persona está retrocediendo:

$Z\ actual < Z\ anterior \rightarrow Avanza;\quad Z\ actual > Z\ anterior \rightarrow Retrocede.$

También se debe analizar si la persona está quieta, entonces se calcula la diferencia entre el Z actual y el Z anterior. Esta diferencia se denomina ΔZ y estará expresada en valor absoluto. Se debe fijar un umbral para la diferencia que aumentará o disminuirá según la necesidad de sensibilidad de la medición. Si ΔZ es menor al umbral definido, entonces la persona se encuentra quieta o detenida. A este monitor lo denominamos Monitor Dinámico porque tiene en cuenta la diferencia entre el estado actual y el anterior. Se puede esquematizar a los monitores y sus características en un diagrama que se muestra en la Fig. 2.

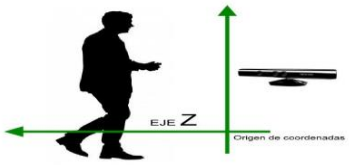

Fig. 1 Eje Z obtenido por Kinect

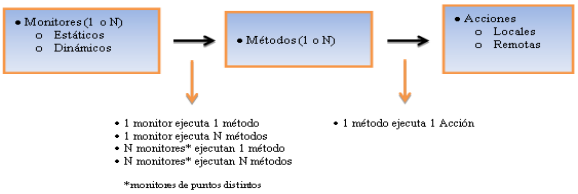

Fig. 2 Esquema de monitores y sus relaciones con otros elementos

El aplicativo internamente generará un string[2] de conexión [16] para comunicarse con el mundo exterior. La estructura de esta cadena ha sido definida arbitrariamente, tal como se observa en la Fig. 3. Donde **Time:** se presenta en segundos, representa el tiempo exacto cuando se realiza la captura. Se incrementará hasta llegar al tamaño de la captura y se repetirá según la cantidad de frames que se capturen. En este caso se repetirá 30 veces antes de pasar al siguiente, **Frame:** Representa la mínima unidad de captura de Kinect. (30 frames por segundo), **X, Y (01):** Valores de X, Y en el joint 01 (Joints del 1 al 20), **X, Y (20):** Valores de X, Y en el joint 20 (Joints del 1 al 20), **M01,…, M02**: Monitores asociados a los joints, **Tamaño Captura:** Tiempo total de captura. (En segundos)A fines prácticos solo se representarán los joints 1 y 20, pero en la captura final todos los joints estarán incluidos.

Time	Frame	X01	Y01	M01	...	X20	Y20	M20	Tamaño Captura

Fig. 3. String de conexión.

5 Framework para interfaz natural de usuario

Se presentan las principales funciones del framework genérico para interfaz humana. En la Fig.4 se muestra un diagrama de clases resumido del desarrollo. Las clases KinectSensor, Joint y Skeleton están relacionadas directamente con las librerías para C# de Kinect. La clase Skeleton contiene una colección de joints y a su vez la clase KinectSensor contiene una colección de Skeletons. La clase Monitor utiliza los métodos de KinectSensor para definir su estado y entregar el mismo. La clase Asignación relaciona al Monitor y al Método y este último ejecuta sus acciones cuando el Monitor este activado. Por último la clase método contiene la secuencia de comandos que se desean realizar. Estos comandos tienen

[2] Se denomina string a una cadena de caracteres ordenada que contiene datos sobre la captura de movimiento de Kinect para ser enviada y utilizada posteriormente

un formato determinado que permite a la Clase Manejador determinar la manera en que serán ejecutados, en otras palabras, que interfaz de salida se utilizará.

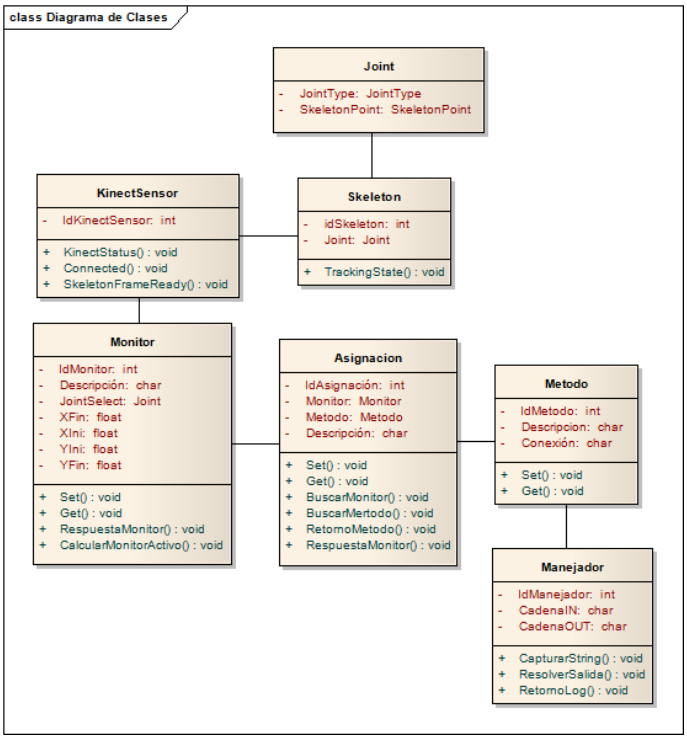

Fig. 4. Diagrama resumido de clases

Se presenta un diagrama de secuencia de la funcionalidad de captura en la Fig. 5. Esta funcionalidad se repetirá continuamente, siempre que el sensor Kinect detecte un skeleton. Primero se obtienen las coordenadas de los puntos del skeleton por medio de la clase KinectSensor. Con estos datos se ejecutan los métodos de la clase Monitor que indican si este se encuentra activo. Cuando se encuentra activo se busca el comando a ser ejecutado en la clase Métodos.
Por último la clase Manejador decide por cuál interfaz de salida será enviado el comando.

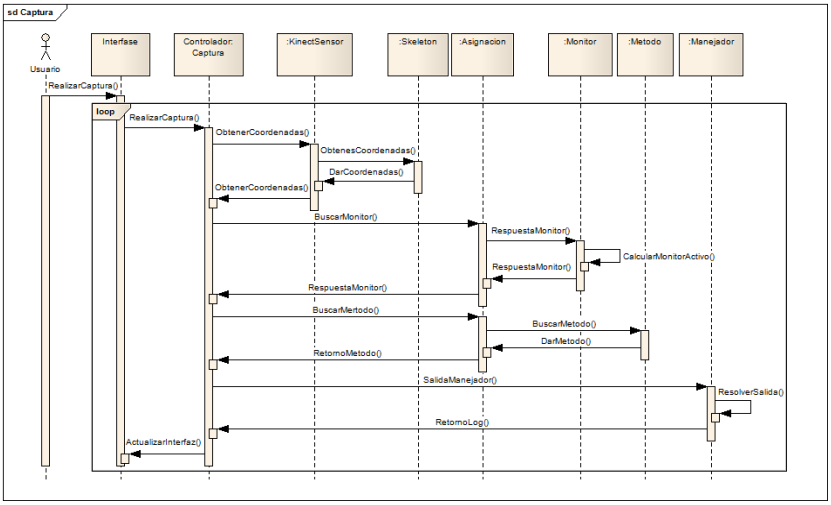

Fig. 5. Diagrama de secuencia de la captura del sensor

La gestión de alta, baja y modificación de monitores, se efectúa a través de una lista desplegable, que permite determinar qué joint será analizado por el monitor. Luego se ingresan los valores iniciales y finales de X e Y por medio de controles deslizantes. En la Fig. 6a, se puede observar una representación del lugar en el plano donde el monitor estará activo, resaltándolo en color rojo. Los valores máximos y mínimos de los controles deslizantes van desde -1 al 1, esto se debe a los datos arrojados por Kinect y a la posición en el que está ubicado; también se tiene en cuenta la posición del usuario frente a él. El sensor ha sido ubicado a una altura de 80 centímetros del suelo y el usuario estará a una distancia de 2 metros del mismo aproximadamente. Se han seleccionado estos valores luego de varias pruebas con el sensor ya que se realizaron muestras de los joints capturados a la distancia anteriormente definida. Para tener una referencia más clara, el valor 1 entregado por Kinect representa 1 metro. Se debe tener en cuenta que el centro de la captura es el origen de coordenadas.

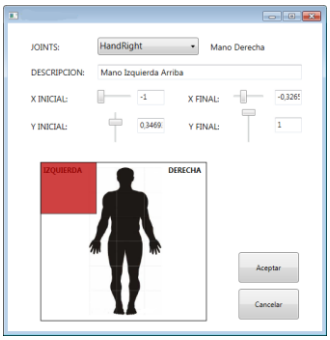

Fig. 6a**.** Creación monitor

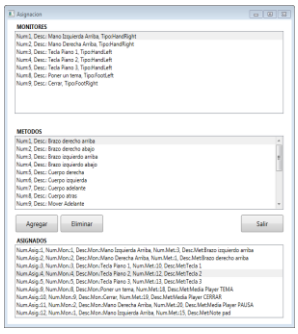

Fig. 6b. Asignación de métodos a monitores

6 Desarrollo de un caso de Aplicación del control de Robots

Se describe un caso de aplicación asociado a la creación de monitores detallados en la sección anterior, aplicando a un robot Robosapien V2 [17], sobre el cual se desarrollaron parte de las pruebas del sistema, en este orden se han agregado métodos predeterminados afines a las acciones del que se pretende controlar. Habiendo generado los monitores, en el siguiente apartado, se relacionará a estos con los métodos que se desean ejecutar, por ejemplo el control a través de la mano derecha arriba. Fig. 6b. Por último se realiza la captura y se indica qué monitores están siendo activados, y por consecuencia qué método se está ejecutando, esto se muestra resaltando en color rojo en la Fig. 7a. Como resultado de esta acción el Robosapien V2 moverá su mano izquierda como se muestra en la Fig. 7b.

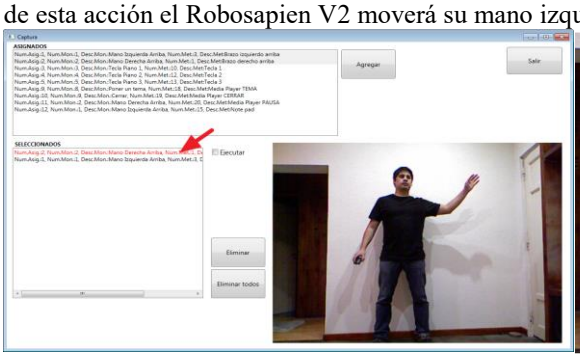

Fig. 7a. Captura de datos

Fig. 7b Robosapien V2 en movimiento.

Habiendo generado el string de conexión detallado en la sección anterior y mediante un proceso de traducción, realizado por un intérprete o interfaz, se envía la orden específica al Robosapien V2 [18] el cual es controlado por medio de comandos IR (infrarrojo) enviados mediante un dispositivo de emisión y recepción de señales infrarrojas, como USB-UIRT [3][19]. Previamente se capturan los principales comandos de movimiento del Robosapien V2 y se relacionan a los métodos de salida de la aplicación. En el momento en que el monitor es activado se pasa a ejecutar el método; la interfaz relacionada, por medio del dispositivo USB-UIRT, trasmite la señal IR al Robosapien V2 y éste realiza la acción indicada (Fig. 8).

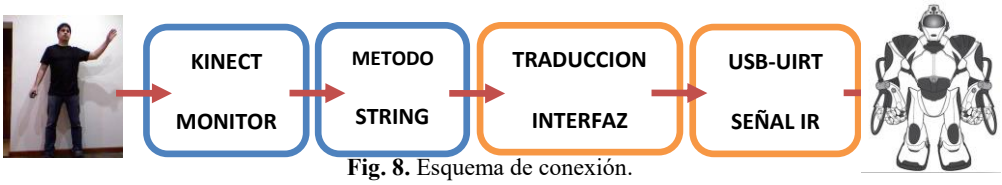

Fig. 8. Esquema de conexión.

[3] Permite a cualquier PC equipado con un puerto USB transmitir y recibir señales infrarrojas hacia cualquier dispositivo multimedia compatible con IR.

El desarrollo del framework se encuentra en la etapa de prototipo de pruebas y fue desarrollado en Microsoft Visual Studio 2010 Express [20] junto con el SDK de Kinect para Windows [21], además se utilizó la librería Usb-Uirt-managed-wrapper [22] para poder utilizar USB-UIRT en C#.
Se puede observar un video donde se presenta una demostración de la aplicación controlando el Robosapien V2 [23]. Se desarrollaron otros casos de prueba de control de un robot donde se pretende que el robot (Roboreptile [24]) se pare en dos patas al realizar el gesto de levantar la mano. Se controla el movimiento lateral de la cabeza del robot cuando el gesto del usuario es el mismo. En la Fig. 9 se pueden ver los movimientos del robot y además se provee un video con la demostración [25].

Fig. 9. Roboreptile en movimiento

7 Conclusiones y futuras líneas de trabajo

La utilización de este framework permite hacer uso de una interfaz natural de forma genérica y genera un bajo acoplamiento con los elementos de su entorno, lo que permite el aprovechamiento del software desarrollado, ya que éste puede ser reutilizado para desarrollos posteriores. Además, los datos de movimiento entregados por el sensor Kinect y los comandos enviados al receptor son totalmente independientes entre sí, permitiendo emitir órdenes de control a diversos dispositivos.
Se destaca que la entrega de datos de control podría realizarse de manera remota, por consiguiente, podría controlarse un dispositivo a grandes distancias, esto se hace posible generando y enviando un string estructurado de conexión.
Las futuras líneas de trabajo se centran en el desarrollo de monitores que faciliten su adecuación en forma dinámica y en este orden también la explotación de información de los monitores bajo un motor de inferencias que explote la información de un motor de reglas para los monitores, métodos y eventos.

8 Referencias

1. Página Oficial de Microsoft Kinect para Windows http://www.microsoft.com/en-us/kinectforwindows/
2. Xbox 360 Pagina official, http://www.xbox.com/es-ES/xbox-360
3. Randomized Decision Forests http://research.microsoft.com/pubs/155552/decisionForests_MSR_TR_2011_114.pdf
4. Main Shift http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/TUZEL1/MeanShift.pdf
5. Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-Time Human Pose Recognition in Parts from Single Depth Images. 2013, 22 de Junio. http://research.microsoft.com/pubs/145347/BodyPartRecognition.pdf
6. Richard Szeliski. (2013, 23 de Junio). Computer Vision: Algorithms and Applications. http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf
7. Perceptive Pixel, http://www.microsoft.com/office/perceptivepixel/
8. Microsoft PixelSense, http://www.microsoft.com/en-us/pixelsense/default.aspx
9. 3D Immersive Touch, http://edusim3d.com/
10. MYO: https://www.myo.com/
11. Ubi Interactive - Kinect touch screen www.ubi-interactive.com
12. Sistema Mirracle http://campar.in.tum.de/WebHome
13. Wiimote and Kinect: Gestural User Interfaces add a Natural third dimension to HCI http://www.infomus.org/Events/proceedings/AVI2012/CHAPTER%203/p116-francese.pdf
14. Robust Hand Gesture Recognition with Kinect Sensor http://eeeweba.ntu.edu.sg/computervision/Research%20Papers/2011/Robust%20Hand%20Gesture%20Recognition%20with%20Kinect%20Sensor.pdf
15. Human Machine Interface HMI using Kinect sensor to control a SCARA http://carlospillajo.info/wp-content/uploads/sites/1369/2014/07/COLCOM2013_Track5-A.pdf
16. MSDN - Sintaxis de cadena de conexión, http://msdn.microsoft.com/es-es/library/ms254500.aspx
17. Robosapien V2 http://wowwee.com/robosapien-x/
18. Manual Robosapien V2 http://www.theoldrobots.com/images7b/Robosapien_V2_Manual.pdf
19. USB-UIRT, http://www.usbuirt.com/
20. Página Oficial de Microsoft Visual Studio 2010 express, http://www.microsoft.com/visualstudio/esn/products/visual-studio-2010-express
21. Kinect SDK, http://www.microsoft.com/en-us/kinectforwindows/develop/learn.aspx
22. JordanZaerr/Usb-Uirt-managed-wrapper, https://github.com/JordanZaerr/Usb-Uirt-managed-wrapper
23. Video caso de prueba Robosapien V2 https://1drv.ms/f/s!At7ZJI6Nbue6ll9N6HsmQ-Dh1u3L
24. Manual Roboreptile http://www.theoldrobots.com/images7b/Roboreptile_Manual.pdf
25. Video caso de prueba Roboreptile https://1drv.ms/f/s!At7ZJI6Nbue6lmDuKmZbn4swtTbu