UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Uso del *Kinect* en el computador desde el punto del vista del usuario

Apellidos, nombre	Agustí i Melchor, Manuel (magusti@disca.upv.es)
Departamento	Departamento de Informática de Sistemas y Computadores (DISCA)
Centro	Escola Tècnica Superior d'Enginyeria Informàtica Universitat Politècnica de València

1 Resumen de las ideas clave

Este documento constituye una **introducción al uso de dispositivos de interacción** complejos denominados **sensores 3D** (o también *depth cameras, Ranging Cameras* o *RGB-D Cameras*). Como ejemplo de uno de estos periféricos que puede utilizar el usuario en computador típico abordaremos la instalación, configuración y uso del *kinect* [1].

Nos referimos a dispositivos que tienen una especial presencia en entornos lúdicos donde la interacción es alta y utiliza diferentes fuentes de información. El interés de estos dispositivos es tanto por su capacidad, como por el abaratamiento de costes que ha supuesto su presencia en el mercado. Dando lugar a la aparición de nuevos dispositivos y aplicaciones, tanto dentro como fuera del contexto de los videojuegos.

Considero interesante dar a conocer el desarrollo que ha dado lugar a estos dispositivos y su forma de funcionamiento. Tanto como dar a conocer las posibilidades de uso que ofrecen las bibliotecas existentes sobre un computador. Ampliando así la capacidad de procesado de la información multimedia de entrada a la que tienen acceso nuestros computadores hoy en día.

2 Objetivos

El presente documento está encaminado a ofrecer una perspectiva inicial de las características propias que un dispositivo del tipo “sensor 3D“ puede ofrecer. Los objetivos de este documento son:

- Que el lector pueda reconocer el abanico de dispositivos de interacción que emplean las plataformas actuales de computadores más allá del ratón y teclado tradicionales.
- Que el lector pueda enumerar las diferentes fuentes de información visual y sonora de este dispositivo, así como también la posibilidad de modificar su orientación vertical y el led de actividad.
- Que el lector sea capaz de enunciar las utilidades del interfaz de OpenKinect.

3 Introducción

En las consolas de videojuegos hemos visto aparecer dispositivos que permiten al usuario interactuar con estos equipos, sustituyendo el uso de los tradicionales teclado y ratón, haciendo más natural la interacción entre el hombre y la máquina. Todos ellos determinan la posición del usuario y la envían al computador junto con otras informaciones como la orientación y aceleración del mismo.

3.1 Evolución de los dispositivos hardware

a) Cámara web. *b) WiiMote.* *c) PS/Move*

Figura 1: Dispositivos predecesores relacionados con los sensores 3D.

Actualmente, existen diversas opciones de dispositivos que permiten la comunicación del hombre con la máquina a través de nuevas formas de interacción:

- En el computador típico de sobremesa , muchas aplicaciones ya incluyen el uso de una cámara de vídeo digital, p. ej. una cámara web, como la de la fig. 1a. Muchas de ellas incorporan un micrófono para obtener también una señal monofónica de sonido. Es la opción más básica y hace recaer en el computador toda la carga de la tarea de interpretar la escena y obtener la información de la interacción a partir del flujo (en escala de grises o color) de vídeo.
- En la consola *Wii* de *Nintendo* han optado por el *WiiMote* [2], fig. 1b. Este implementa la interacción con una barra de leds infrarrojos (que se sitúa debajo del monitor, p. ej.) y un dispositivo que lleva el usuario (formado por una cámara de vídeo y otros sensores de posición y aceleración). Este dispositivo triangula su posición respecto a la barra que permanece fija.
- En el caso de la *PlayStation* de *Sony*, en el *PS/Move* [3] la interacción se obtiene a partir de una cámara (que se suele dejar debajo del monitor) de vídeo digital, que analiza la escena buscando el dispositivo, fig. 1c, que lleva el usuario (formado por una bola de color configurable y otros sensores de posición y aceleración).
- La *Xbox* de *Microsoft* [1] por su parte ha optado por un dispositivo que había realizado la empresa (que es ahora propiedad de *Apple*[1]) *PrimeSense*, el *kinect*. Con este dispositivo es posible plantear una interacción más natural, en tanto que son las acciones del cuerpo del propio usuario las que se convierten en eventos para el computador. Este dispositivo está constituido por una cámara de vídeo, una cámara de infrarrojos, micrófonos y un motor. Estas "cámaras" proporcionan, fig. 2, dos fuentes de información visual sincronizadas: una información de color (imagen RGB) sincronizada con información, para cada píxel, de distancia al sensor (*depth image*). También es habitual que permitan capturar la información de audio de alrededor mediante uno o varios micrófonos.

[1] Shel Israel. (Nov. 2013),"Why Apple Bought PrimeSense" <http://www.forbes.com/forbes/welcome/>.

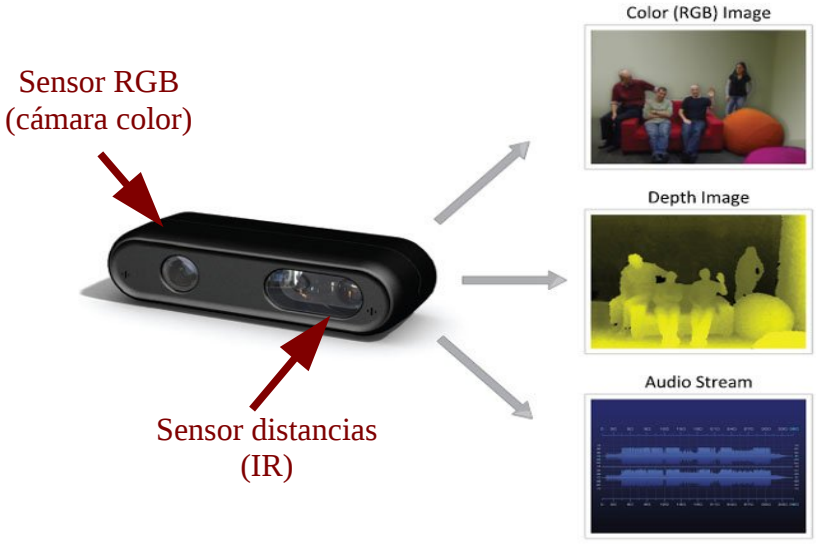


Figura 2: Tipos de información de entrada habituales en un sensor 3D. Imagen extraída de <http://www.hizook.com/files/users/3/PrimeSense_DepthCamera.jpg>.

Figura 3: Ejemplos de sensores 3D: a) Xtion, b) RealSense 3D y c) Structure Sensor.

Hoy en día encontramos muchos ejemplos de **sensores 3D** tanto dentro como fuera del contexto de los videojuegos. La *fig. 3* muestra[2], aparte del kinect, otros ejemplos de sensores de tipo RGB-D indicando los componentes característicos (ya enunciados) de este tipo de dispositivos. Agrupándolos en función del tamaño del sensor, podemos encontrar:

- El *Xtion* de Asus, también creado en colaboración con *Prime Sense* y el *RealSense 3D* de Intel y *Creative*, Para equipos de sobremesa.
- El *Structure Sensor*[3] de *Occipital, p*ara tabletas y dispositivos móviles.

[2]Imágenes tomadas respectivamente de <https://www.asus.com/3D-Sensor/Xtion_PRO_LIVE/>, <https://software.intel.com/en-us/blogs/2015/01/26/can-your-webcam-do-this> y <http://structure.io/>.

[3] Véase la página web del fabricante en <http://structure.io/>.

- Las que se están dando en llamar *Smartglasses* (como una de las piezas claves de estos "visores") como[4] las *SpaceGlasses* de *Meta*, *Google Glass*, *Atheer*, o las *HoloLens* de *Microsoft*, por citar unas algunas.

3.2 Proyectos existentes

Estos sensores 3D se conectan al computador mediante el uso de una serie de aplicaciones que proporcionan el acceso al interfaz hardware del dispositivo, así como también el interfaz de desarrollo de aplicaciones que los manejen (conocido también por las siglas SDK). De manera cronológica, han ido apareciendo más o menos así:

- En Noviembre de 2010, días después de llegar el *kinect* al mercado, *libfreenect (el resultado del trabajo de H. Martín[5] para el concurso que lanza Adafruit)* aparece en escena[6] y da pie *OpenKinect* como un proyecto abierto que ofrece un *driver* (la aplicación de bajo nivel que hace de interfaz entre el computador y un periférico) así como un conjunto de funciones para el desarrollador.
  También aparece una primera versión[7] del grupo NUI, especialistas en proyectos relacionados con interfaces multitáctiles.
- En Diciembre de 2010, el proyecto *OpenNI* ve la luz a partir de que *PrimeSense* libera sus *drivers* y una biblioteca de funciones capaz de detección de movimiento (NITE).
- En Febrero de 2012 aparece la primera versión del driver y el soporte de ejecución para un computador por parte de *Microsoft*. En Mayo aparece la primera versión del SDK.
- En 2012 *Prime Sense* y Asus unirán fuerzas par crear un dispositivo más pequeño en tamaño y consumo, el *Wavi Xtion*.
- En Noviembre de 2013, aparece el proyecto *OpenNI2* propiciado por Occipital (junto con el desarrollo del *Structure Sensor*) tras la adquisición de *PrimeSense* por *Apple* y el cierre del sitio web de OpenNI.

De todos estos proyectos solo *OpenKinect* y *OpenNI2*, a fecha de la realización de este trabajo continúan activos y son de naturaleza multiplataforma. Así que dejaremos los otros a un lado y, de estos dos, vamos a profundizar en *OpenKinect / libfreenect* por que permite el acceso a un lkinect virtual o simulado.

3.3 Conexión al computador

Para conectar el dispositivo con el computador es necesario disponer de la conexión física y lógica. La conexión física es la de los cables y conectores que permiten conectar el ordenador con el *kinect*, lo cual es más o menos directo, ya que existen diferentes modelos de este dispositivo.

---

[4] En el orden en que son citadas, las páginas web de los fabricantes son: <https://www.getameta.com>, <https://developers.google.com/glass/>, <http://atheerair.com/> y <https://www.microsoft.com/microsoft-hololens/en-us/>.

[5] También conocido como "marcan", tiene su sitio web en <https://marcan.st/about/>.

[6] Los detalles los puede encontrar en "Inside the race to hack the Kinect", disponivle en <https://www.newscientist.com/article/dn19762-inside-the-race-to-hack-the-kinect/>.

[7] En la página web del grupo <https://codelaboratories.com/kb/nui>, se describe como una versión preliminar

Figura 4: Conexión de alimentación y USB.

En los primeros modelos que aparecieron en el mercado (1414, 1473 y el 1517), que se vendían junto a la consola, necesitan un adaptador para poder conectar el dispositivo a un puerto USB y, directamente, a la alimentación. Se necesita el adaptador para conectar estos modelos al computador. El *kinect* que se vende con la consola no lo trae, el que se vende solo sí. Así que cuidado si el gusanillo te pica ... ¡mira el cable!.

Con el dispositivo conectado als USB 2.0 y a la corriente eléctrica (fig. 4), la luz del dispositivo parpadea y la del conversor (o adaptador) está encendida fija, indicando actividad. La segunda versión denominada "Kinect for Windows version 2" (aparecido allá por Noviembre de 2013) ha ampliado su *hardware*.

Feature	Kinect for Windows 1	Kinect for Windows 2
Color Camera	640 x 480 @30 fps	1920 x 1080 @30 fps
Depth Camera	320 x 240	512 x 424
Max Depth Distance	~4.5 M	~4.5 M
Min Depth Distance	40 cm in near mode	50 cm
Horizontal Field of View	57 degrees	70 degrees
Vertical Field of View	43 degrees	60 degrees
Tilt Motor	yes	no
Skeleton Joints Defined	20 joints	26 joints
Full Skeletons Tracked	2	6
USB Standard	2.0	3.0
Supported OS	Win 7, Win 8	Win 8
Price	$299	TBD

Figura 5: Tabla comparativa de los modelos 1 y 2 de kinect.

La fig, 5 muestra[8] los cambios de forma comparada: observe como ha mejorado la resolución de las cámaras y la capacidad del procesador interno que devuelve ahora un mayor número de puntos y puede hacer el seguimiento de una mayor número de esqueletos (por tanto de personas). A cambio ha eliminado el motor que en las primeras versiones permitía el "cabeceo" del dispositivo ¿Se habrá caído alguno? Esta versión puede conectarse directamente al USB.

La conexión lógica es la que implementa el interfaz para acceso a la información que suministra el dispositivo desde una aplicación y se divide en bajo y alto nivel. La de bajo nivel implementa el manejador (*driver*) y es, generalmente, proporcionado por el fabricante. Esta "conexión" permite (si el núcleo del operativo al que se le conecta no reconoce el dispositivo) acceder a los datos en bruto que suministra el dispositivo (imagen o audio); así como a los registros que permiten su configuración (color y parpadeo del *led* y movimiento del motor).

Sobre esta conexión mínima, una biblioteca de funciones de alto nivel ofrece un conjunto de operaciones de mayor complejidad como detección de gestos, el esqueleto, caras, etc. que facilitan el desarrollo de aplicaciones abstrayendo los detalles de bajo nivel de acceso al dispositivo. Para implementar este interfaz existen dos grandes alternativas de carácter multiplataforma: *OpenKinect* [4] /libfreenect.y *OpenNI2*[9] */ OpenNI (NITE+ Prime Sense)*. Las hemos enunciado en el punto 3.2 Proyectos existentes y las describiremos con detalle en el punto 4 Desarrollo.

4 Desarrollo

A la hora de **iniciar** el desarrollo es importantísimo instalar un SDK y, quizá, un manejador que permita el acceso por parte del núcleo del operativo al *hardware* del dispositivo. Revisar la Instalación que hemos llevado a cabo y su configuración es necesario para disponer de un punto de partida sobre el que poder crear nuevas aplicaciones y nos servirá para explorar las posibilidades del dispositivo con las utilidades que obtenemos al instalar el *software* de soporte.

En nuestro caso, hemos decidido utilizar *OpenKinect/libfreenect*, por la opción de *Fakenect*, que ahora explicaremos. *libfreenect* es una blibioteca que ofrece funciones para acceder al *Microsoft Kinect* a través de la conexión USB. Permite el acceso a la cámara RGB así como a la de profundidad, el motor, los acelerómetros y el *led* de actividad. *OpenKinect* es el proyecto en que se engloba esta librería de operaciones muy bajo nivel y sobre la que están desarrollando otras más elaboradas como la típica detección del esqueleto y de los gestos de la mano.

4.1 Instalación y uso de *OpenKinect* (*libfreenect*)

Sugiero recurrir primer a la instalación que viene preparada en los repositorios del operativo instalado. En el caso de Ubuntu (versión 11.10) y Debian (ver. 7), exsiten estos paquetes precompilados que se pueden instalar simplemente con la orden:

```
$ sudo apt-get install freenect
```

[8] Extraída de "How Does The Kinect 2 Compare To The Kinect 1?" <http://zugara.com/how-does-the-kinect-2-compare-to-the-kinect-1>.

[9] Tras la compra de Prime Sense por parte de Apple y el cierre del sitio original de OpenNI (http://www.openni.org/), otros fabricantes de aplicaciones y dispositivos basados en las mismas ideas mantienen la nueva versión denominada OpenNI2 en <http://structure.io/openni>.

Existe una modificación de F. Echtler que permite quitar el *driver* de memoria si no está conectado el dispositivo, pero si observa que después de ejecutarlo por primera vez ya no puede volver a hacerlo una segunda es por que tiene una versión ligeramente anterior y ha de hacer[10], en la consola:

```
$ sudo modprobe -r gspca_kinect
$ sudo modprobe -r gspca_main
$ echo "blacklist gspca_kinect" |sudo tee -a /etc/modprobe.d/blacklist.conf
```

El dispositivo está disponible para cualquier usuario del grupo 'plugdev'. Se puede comprobar si ya está conectado (y reconocido por el sistema) con la orden *lsusb*, cuyas tres últimas líneas de información devuelta, deben mostrar que se han asignado los dispositivos a los recursos del *kinect*:

```
$ lsusb
...
Bus 001 Device 004: ID 045e:02b0 Microsoft Corp. Xbox NUI Motor
Bus 001 Device 005: ID 045e:02ad Microsoft Corp. Xbox NUI Audio
Bus 001 Device 006: ID 045e:02ae Microsoft Corp. Xbox NUI Camera
```

4.2 Utilidades instaladas

Entre las demos se encuentran tres aplicaciones básicas para observar el fundionamiento del Kinect: *freenect-cppview*, *freenect-glpclview* y *freenect-glview*. Además es interesante la existencia de *record* y *fakenect [8]*.

4.2.1 Utilidades de acceso a OpenKinect

En la instalación de *libfreenect* vienen tres aplicaciones que vamos a describir, La más básica es *freenect-glview*, la fig. 6 muestra su salida en pantalla. Pulsando la tecla 'f' se va cambiando el modo de vídeo:de RGB (fig. 6a) a la imagen de la cámara de profundidad (infrarrojos, fig.6b). Observe con en las imágenes de la izquierda, de la fig. 6, se muestra coloreada la imagen de profundidad en base a intervalos de los valores ofrece la cámara de profundidad. Los valores en negro indican que el procesador interno no ha sido capaz de determinar la distancia para esos puntos, por caer fuera del rango de trabajo: demasiada cercanía o lejanía.

Por otro lado, *freenect-cppview*,[11], permite el acceso al led de actividad del frontal del dispositivo y al motor. Lo que ofrece la posibilidad de: cambiar el color del *led*, respectivamente con '1', '2', '3'. '4'[12], '6' y '0', entre verde, amarillo, verde parpadeando, rojo y amarillo alternativamente y apagado; activar el motor para cambiar la posición vertical del dispositivo (dentro de un ángulo de ±30º), incrementado con 'w', decrementando con 's''[13], e inicializándolo a cero con 'x', a 10º con 'e' y a -10º con 'c'.

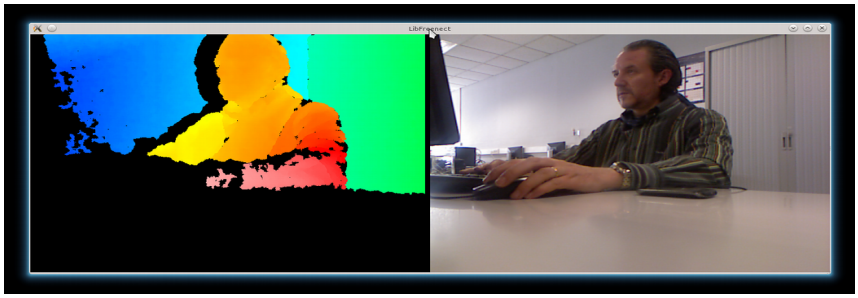
---

[10] Getting Started <https://openkinect.org/wiki/Getting_Started>.

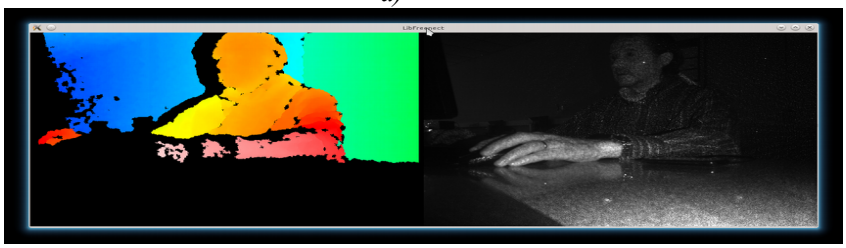
[11] En la URL <https://github.com/OpenKinect/libfreenect/blob/master/wrappers/cpp/cppview.cpp>.

[12] Pulsar el '5' da el mismo resultado que '4'.

[13] También con 'd'.



a)



b)

Figura 6: Salida gráfica de freenect-glview el modo a)RGB y b) distancias.

Para acabar, *freenect-glpclview*[14], ofrece una mezcla las dos imágenes para hacer un alzamiento 3D de la escena con la textura de la imagen RGB sobrepuesta. La fig. 7a muestra la salida de la aplicación en su inicio y como se ha modificado la perspectiva durante la ejecución, mediante el ratón, lo que permite ver los objetos "separados" (fig. 7b) y el fondo de la escena, en función de valores predeterminados de rangos de profundidad.

Aquí se pueden utilizar las teclas 'w' y 's' para aumentar o decrementar el zoom de la imagen. También la tecla 'c' para activar la textura o no.

Figura 7: Salida gráfica de la utilidad freenect-glpcview: a) Inicial y b) Durante la ejecución, mediante el ratón

4.2.2 *Fakenect* y *record*

Es especialmente interesante puesto que posibilita simular el interfaz del dispositivo físico y ejecutar aplicaciones para él sin tenerlo físicamente

[14] En la URL <https://github.com/OpenKinect/libfreenect/blob/master/examples/glpclview.c>.

conectado[15]. Y para probarlo puede utilizar uno de los tres conjuntos de datos disponibles *[8]* oficialmente: *legos*, *sophie* o *thanksgiving*. Para utilizarlos hay que ejecutar la aplicación, tal cual se ha creado para el dispositivo físico, con una orden al estilo de:

```
$ LD_PRELOAD="/usr/lib/fakenect/libfreenect.so" FAKENECT_PATH="thanksgiving0"
/usr/bin/freenect-glview
```

donde:

- *LD_PRELOAD* carga una versión de la biblioteca dinámica de funciones que se encargará de traducir los accesos a hardware del dispositivo por accesos a ficheros locales. Tiene el mismo nombre que la que originalmente se utilizaría, por lo que no cargará y la emulación se hace de forma transparente a la aplicación final.
- *FAKENECT_PATH* es la ruta a un directorio donde encontrar esos ficheros que se van a utilizar durante la simulación
- y el último elemento de la orden es la ruta a la aplicación a ejecutar.

Si tiene acceso al un *kinect* real puede grabar su propia secuencia y trabajar con ella posteriormente en este modo. Simplemente deberá ejecutar la orden[16] *record*.

5 Conclusión

A lo largo de este documento, hemos explorado básicamente el uso de la información multimedia que proporciona un sensor 3D como el *kinect* porqué es ejemplo actual de dispositivo para la interacción entre hombre y máquina.

Tras la lectura detenida de este documento se estará en condiciones de iniciar la experimentación con las herramientas ya construidas que ofrece el API de OpenKinect/libfreenect y otros ejemplos disponible en el GITHub del proyecto1[17], como son *camtest.c, tiltdemo.c* (para el acceso al motor), o *micview.c* y *wavrecord.c* (para acceder al registro de audio a través del vector de micrófonos disponible). Así como también se estará en situación de probar las que se pueden encontrar en la red.

De este modo el lector tienen opciones para evaluar un dispositivo y una aplicación para poner en marcha una determinada idea Y ya veremos que más nos depara el futuro. Ánimo y a experimentar.

6 Bibliografía

[1] Kinect for Windows Sensor Components and Specifications. Disponible en <https://msdn.microsoft.com/en-us/library/jj131033.aspx>.
[2] Brain,M. (2007). How the Wii Works" Disponible en HowStuffWorks.com. <http://electronics.howstuffworks.com/wii.htm> Consultada el 1 de Junio de 2015.

[15] Try Kinect apps without a Kinect thanks to Fakenect

http://awesomebytes.com/2010/11/27/prueba-aplicaciones-para-kinect-sin-un-kinect-gracias-a-fakenect/

[16] Que deberá descargar de <https://github.com/OpenKinect/libfreenect/tree/master/fakenect>.

[17] En la URL <https://github.com/OpenKinect/libfreenect/tree/master/examples>..

[3] Valdes, R. (2005). How PlayStation 3 Works Disponible en HowStuffWorks.com. <http://electronics.howstuffworks.com/playstation-three.htm> Consultada el 1 de Junio de 2015.
[4] OpenKinect. Disponible en: <https://openkinect.org/wiki/Main_Page>.
[5] Bloisi, D. d. y Pennisi, A. (2015). Robot Programming. Section of Elective in Artificial Intelligence. Master Artificial Intelligence and Robotics. Department of Computer, Control and Management Enginneering Antonio Ruberti. Sapienza Università Di Roma. Disponible en <http://www.dis.uniroma1.it/~nardi/Didattica/CAI/matdid/image_processing_with_opencv.pdf>.
[6] Billie, G. (2011). Microsoft Kinect Sensor Evaluation NASA USRP. Internship Final Report. Johnson Space Center. 8/5/2011. Disponible en <http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20110022972.pdf>.

[7] PrimeSense™ 3D Sensors. Disponible en <www.i3du.gr/pdf/primesense.pdf>.
[8] Fakenect. Disponible en <https://openkinect.org/wiki/Fakenect>.