INTEGRACIÓN A MEDIOS DIGITALES
2K17/2K18

MEMORIA

CONTROL DEL CURSOR
MEDIANTE
ASUS XTION PRO LIVE

ALUMNO:

DAVID BERNABÉ TORRES

ÍNDICE

1. PRESENTACIÓN DEL PROYECTO

El proyecto planteado se centra en el control del cursor mediante la cámara ASUS XTION PRO, sustituyendo así el uso de un ratón.

En esta memoria presentaremos el material utilizado para la realización del proyecto (librerías, versiones, entorno de programación), su ejecución y el modo de empleo, así como el estudio realizado del código para facilitar su posterior uso por otra persona.

2. OBJETIVOS

Controlar el cursor mediante el reconocimiento de nuestra mano por parte del ASUS XTION PRO y realizar acciones tales como pulsar, arrastrar o ampliar y disminuir imágenes mediante una serie de movimientos con la mano que el sensor es capaz de leer, analizar e interpretar.

Una vez detectados esos movimientos con la ayuda de OpenNi+NITE, será posible llamar a los eventos del sistema encargado de realizar las acciones en Windows (dichos eventos no están pensados para ser realizados en otro sistema).

3. ASUS XTION PRO LIVE

Xtion PRO LIVE[1] emplea un sensor de infrarrojos, la detección adaptativa de la profundidad, el color y sonido para capturar los movimientos del usuario en tiempo real, movimiento y la voz, para detectar a los usuarios con más precisión. Esta solución integra unas herramientas para facilitar la creación de aplicaciones basadas en movimiento sin necesidad de hacer uso de complicados algoritmos.

1. **Detección de gestos:** la solución Xtion PRO detecta el movimiento de las manos sin retraso alguno, lo que permite emplearlas como controlador, permitiendo empujar, seleccionar, hacer movimientos circulares, ondas y mucho más, ideales para emplear como interfaz para cualquier tipo de aplicación.

2. **Detección del cuerpo entero:** ideal para el desarrollo de juegos.

3. **RGB**: Xtion PRO LIVE capta la imagen del usuario, lo que es útil para la detección de personas, señalización digital, sistemas de seguridad y otro tipo de aplicaciones.

4. **Audio:** La capacidad de reconocer sonidos permite que el usuario controle todo tipo de parámetros con su voz. Esta funcionalidad junto con RGB permite llevar a cabo videoconferencias.

Además, Xtion PRO LIVE también es compatible con Unity3D para permitir un desarrollo de juegos y aplicaciones más simple

[1]Imagen cámara ASUS XTION PRO LIVE

4. ESPECIFICACIONES DEL ENTORNO DE TRABAJO

Para el desarrollo del trabajo realizado se ha empleado diferente software y librerías que nos han hecho posible partir de una base con la que empezar el proyecto y la realización del mismo. A continuación, se detalla las versiones empleadas y los enlaces de descarga para su uso:

- Nite2.2. La utilización de Nite nos ha facilitado el reconocimiento de las manos y la creación de un puntero que lo identifica (ejemplo HandViewer), base muy importante para el proyecto realizado. Por otro lado, tenemos más ejemplos tales como reconocimiento de cuerpo, lectura de datos, etc.

  Para la instalación de Nite accedemos al enlace proporcionado en este apartado, seleccionamos nuestro sistema operativo, abrimos el '.zip' y ejecutamos el instalador.

  Para poder probar los ejemplos necesitaremos disponer de una cámara como la mostrada en la imagen anterior[1] y acceder a "PrimeSense -> NiTe2 -> Samples -> Bin" donde encontraremos todos los ejecutables.

  Enlace->https://bitbucket.org/kaorun55/openni-2.2/src/2f54272802bf/NITE%202.2%20%CE%B1/?at=master

- OpenNi2. El uso de OpenNi nos ha proporcionado la capacidad de poder reconocer la cámara a emplear y la captura y procesado de las imágenes para su posterior tratamiento con Nite para el reconocimiento de manos.

  Para la instalación de OpenNi2 accedemos al enlace proporcionado en este apartado, seleccionamos nuestro sistema operativo, abrimos el '.zip' y ejecutamos el instalador.

  Para poder probar los ejemplos necesitaremos disponer de una cámara como la mostrada en la imagen anterior[1] y acceder en este caso a "OpenNI2 -> Samples -> Bin" donde encontraremos todos los ejecutables.

  Enlace -> https://structure.io/openni

5. CÓDIGO EMPLEADO Y FUNCIONALIDAD

En este apartado nos centraremos en la funcionalidad añadida al ejemplo base tomado HandViewer, que únicamente nos mostraba una ventana con la imagen que capturaba y donde se mostraba cuando detectaba una mano añadiéndole un puntero a esa mano detectada, y un terminal que nos servía para visualizar cuando encontraba una mano, cómo la identificaba y cuando la perdía.

Así pues, podemos resumir el código implementado con el siguiente esquema y pasos seguidos:

1. Reconocimiento y borrado de manos
2. Acciones a realizar si detectamos una mano
   a. Mover el cursor por la pantalla
   b. Cuando hacer un 'click izquierdo'
3. Acciones a realizar si detectamos una segunda mano
   a. Calcular valores segunda mano
   b. Cuando hacer un 'doble click' y un 'click derecho'

1. Reconocimiento y borrado de manos

En esta primera parte, disponemos de la imagen[2] con el código implementado que realiza la detección de las manos. Para poder seguir utilizando el programa aunque se pierdan y reconozcan varias manos, hemos creado un array en la que guardamos el ID de las manos al detectarlas y las borramos del array en el momento en el que se pierde la mano. De esta forma, la primera mano detectada se almacenará en la primera posición y será la encargada de controlar el cursor y hacer '*click izquierdo',* mientras la segunda mano se almacenará en la segunda posición y se encargará de acciones como '*click derecho'* o '*doble click'.*

```
const nite::Array<nite::HandData>& hands= handFrame.getHands();
for (int i = 0; i < hands.getSize(); ++i)
{
    const nite::HandData& user = hands[i];

    if (!user.isTracking())
    {
        printf("Lost hand %d\n", user.getId());

        //si la mano se ha perdido, vamos a borrarla del array que la identifica
        for (int i = 0; i < sizeof(manos); i++) {

            if (manos[i] == user.getId()) {
                manos[i] = NULL;
                break;
            }
        }
        nite::HandId id = user.getId();
        HistoryBuffer<20>* pHistory = g_histories[id];
        g_histories.erase(g_histories.find(id));
        delete pHistory;
    }
    else
    {
        if (user.isNew())
        {
            printf("Found hand %d\n", user.getId());

            //si encontramos una mano, la añadimos al espacio en nulo que hay
            for (int i = 0; i < sizeof(manos); i++) {
                if(manos[i]==NULL){
                    manos[i] = user.getId();
                    break;
                }
            }
            g_histories[user.getId()] = new HistoryBuffer<20>;
        }
```

[2] Código detección de manos

2. Acciones a realizar si detectamos una mano

Mover el cursor por la pantalla

En el siguiente apartado encontramos la imagen[3] con el código implementado encargado de una vez detectada la primera mano, aplicarle una serie de acciones.

Hemos realizado una media de las primeras cinco coordenadas que tiene la mano, para evitar así una vibración molesta del cursor en el momento que llamemos al evento 'SetCursorPos()', encargado de mover el cursor del ordenador en base a dos variables que referenciaran a las posiciones X e Y de la pantalla.

Un aumento considerable del número de valores con los que se hace la media podría dar lugar a una visualización del movimiento del cursor 'a saltos'.

```
// Add to history
HistoryBuffer<20>* pHistory = g_histories[user.getId()];

//si hay una mano encontrada en la posición zero(por lo general la primera encontrada)
if (manos[0]==user.getId()) {

    //le añandimos un puntero
    pHistory->AddPoint(user.getPosition());

    //sumamos los valores de posición en el eje X e Y
    sumaX += user.getPosition().x;
    sumaY += user.getPosition().y;
    contador++;

    //si hemos sumado ya 5 valores de X e Y
    if (contador == N) {

        //reseteamos el contador y calculamos la media
        contador = 0;
        sumaX = sumaX / N;
        sumaY = sumaY / N;

        //acumulamos la media y guardamos la ultima referencia
        valorAnteriorx += sumaX;
        valorAnteriory += sumaY;
        ultimaReferenciax = sumaX;
        ultimaReferenciay = sumaY;
        contadorClick++;

        //hacemos que el cursor se mueva al vamor medio calculado (multiplicados para poder moverse por toda la pantalla)
        SetCursorPos((int)sumaX * 4, (int)sumaY*(-5));

        //reseteamos suma
        sumaX = 0;
        sumaY = 0;
    }
```

[3] Mover el cursor por la pantalla

Cuando hacer un ‘click izquierdo’

En este apartado continuación del anterior, hemos realizado un contador para que cuando realice 4 sumas de las medias, compruebe si esos valores han sido muy cercanos y no ha habido mucha variación para así detectar si la mano ha estado parada. Si es así, llamamos a la función ‘*mouse_event’[4]* (necesario incluir “windows.h”) para realizar un ‘*click izquierdo’* y seguidamente reseteamos los valores que tenemos. Si hemos movido demasiado la mano, simplemente reseteamos los valores.

```
        //si ya tenemos la suma de 4 medias
        if (contadorClick == 4) {

            //comprobamos si los valores medios distan de la ultima referencia, es decir, si tenemos la mano quieta
            if (((valorAnteriorx / 4) - ultimaReferenciax) <= 1 &&
                ((valorAnteriory / 4) - ultimaReferenciay) <= 1 &&
                ((valorAnteriorx / 4) - ultimaReferenciax) >= -1 &&
                ((valorAnteriory / 4) - ultimaReferenciay) >= -21) {

                //llamamos a la funcion de windows.h para hacer un click y reseteamos valores
                mouse_event(MOUSEEVENTF_LEFTDOWN, (int)ultimaReferenciax * 4, (int)ultimaReferenciay*(-5), 0, 0);
                mouse_event(MOUSEEVENTF_LEFTUP, (int)ultimaReferenciax * 4, (int)ultimaReferenciay*(-5), 0, 0);
                valorAnteriorx = 0;
                valorAnteriory = 0;
                ultimaReferenciax = 0;
                ultimaReferenciay = 0;
                contadorClick = 0;
            }
            else {

                //si la mano no esta quieta, reseteamos valores
                valorAnteriorx = 0;
                valorAnteriory = 0;
                ultimaReferenciax = 0;
                ultimaReferenciay = 0;
                contadorClick = 0;
            }
        }
```

[5] Hacer un ‘*click izquierdo’*

3. Acciones a realizar si detectamos una segunda mano

Calcular valores segunda mano

Ahora pasamos a realizar acciones si hay una segunda mano detectada. En este apartado simplemente cogemos sus valores en el eje X e Y, y realizamos las medias de la posición para utilizarlo más adelante (imagen [6])

```
//si hay una mano en la posición 1 de manos(por lo general la segunda mano)
if (manos[1]==user.getId()) {

    //Obtengo los datos de la mano y le asigno un puntero
    HistoryBuffer<20>* pHistory = g_histories[user.getId()];
    pHistory->AddPoint(user.getPosition());

    //sumamos los valores de posición en el eje X e Y de la segunda mano
    sumaXM2 += user.getPosition().x;
    sumaYM2 += user.getPosition().y;
    contadorMano2++;

    //si hemos sumado ya 5 valores de X e Y
    if (contadorMano2 == N) {

        //reseteamos el contador y calculamos la media
        contadorMano2 = 0;
        sumaXM2 = sumaXM2 / N;
        sumaYM2 = sumaYM2 / N;

        //acumulamos la media y guardamos la ultima referencia
        valorAnteriorxM2 += sumaXM2;
        valorAnterioryM2 += sumaYM2;
        ultimaReferenciaxM2 = sumaXM2;
        ultimaReferenciayM2 = sumaYM2;
        contadorClickM2++;

        //reseteamos suma
        sumaXM2 = 0;
        sumaYM2 = 0;
    }
```

[6] Cálculos datos segunda mano

Cuando hacer un *'doble click'* y un *'click derecho'*

En la imagen[7] apreciamos como calculadas ya las medias y los valores de interés, pasamos a realizar una serie de eventos según estos datos. Para ello, se ha decidido que si se calcula que hemos movido bruscamente la posición de la segunda mano, hacemos un '*click derecho',* que nos abrirá el menú contextual del elemento donde se encuentre el cursor que controlamos con la primera mano. Por otro lado, estamos analizando constantemente si se realiza algún gesto con la mano. Estos gestos ya vienen definidos en Nite y son:

- GESTURE_WAVE: Detecta el gesto si agitamos rápidamente la mano.
- GESTURE_CLICK: Detecta el gesto si simulamos un 'click' con la mano.
- GESTURE_HAND_RAISE: Mantener la mano levantada hacia la cámara (este gesto es el que más cuesta de ser detectado por la cámara)

Al detectar alguno de estos gestos nombrados, realizaríamos un '*doble click'* para realizar acciones tales como ampliar una ventana, minimizarla o abrir carpetas.

```
    //si ya tenemos la suma de 4 medias
    if (contadorClickM2 == 4) {
        //printf("Valor de la resta valor anterior y ultima referencia es %f\n", ((valorAnteriorxM2 / 4) - ultimaReferenciaxM2));

        //comprobamos si los valores medios distan de la ultima referencia para saber si hemos movido
        if (((valorAnteriorxM2 / 4) - ultimaReferenciaxM2)>=60 || ((valorAnteriorxM2 / 4) - ultimaReferenciaxM2)<= -60) {

            //si hemos movido suficiente la mano, realizaremos un click derecho típico de ratón
            mouse_event(MOUSEEVENTF_RIGHTDOWN, (int)ultimaReferenciax * 4, (int)ultimaReferenciay*(-5), 0, 0);
            mouse_event(MOUSEEVENTF_RIGHTUP, (int)ultimaReferenciax * 4, (int)ultimaReferenciay*(-5), 0, 0);

            //reeteamos valores
            valorAnteriorxM2 = 0;
            valorAnterioryM2 = 0;
            ultimaReferenciaxM2 = 0;
            ultimaReferenciayM2 = 0;
            contadorClickM2 = 0;
        }
        else {

            //si tenemos la mano quieta, reseteamos valores
            valorAnteriorxM2 = 0;
            valorAnterioryM2 = 0;
            ultimaReferenciaxM2 = 0;
            ultimaReferenciayM2 = 0;
            contadorClickM2 = 0;
        }
    }

    //si en algún momento hacemos algún gesto(definidos por Nite, como pulsar) entonces hacemos doble click
    for (int i = 0; i < gestures.getSize(); ++i)
    {
        if (gestures[i].isComplete())
        {
            mouse_event(MOUSEEVENTF_LEFTDOWN, (int)ultimaReferenciax * 4, (int)ultimaReferenciay*(-5), 0, 0);
            mouse_event(MOUSEEVENTF_LEFTUP, (int)ultimaReferenciax * 4, (int)ultimaReferenciay*(-5), 0, 0);
            mouse_event(MOUSEEVENTF_LEFTDOWN, (int)ultimaReferenciax * 4, (int)ultimaReferenciay*(-5), 0, 0);
            mouse_event(MOUSEEVENTF_LEFTUP, (int)ultimaReferenciax * 4, (int)ultimaReferenciay*(-5), 0, 0);
            break;
        }
    }
}
```

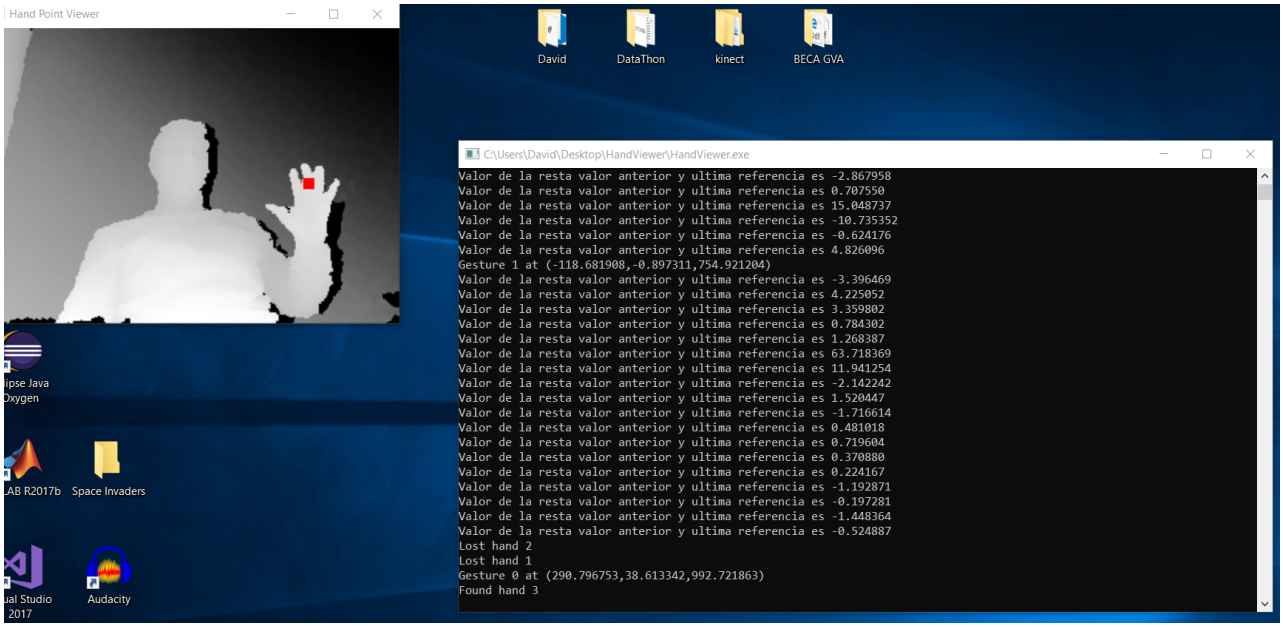
[7] Realización de *'click derecho'* y *'doble click'*

6. MODO DE USO

Antes de todo, es importante decir que la primera mano detectada será la que controle el cursor, así que si eres zurdo, realiza primero la detección de tu mano izquierda o al contrario en caso de ser diestro. Por otro lado, la cámara no detecta bien las imágenes al estar muy cerca, por lo que se recomienda estar a una cierta distancia de ella para poder hacer las animaciones y las detecciones correctamente. Ahora pasamos al uso de esta aplicación.

1º. Detectar mano.

Colóquese en dirección a la cámara a una distancia de un metro y agite la mano para ser detectada [8].



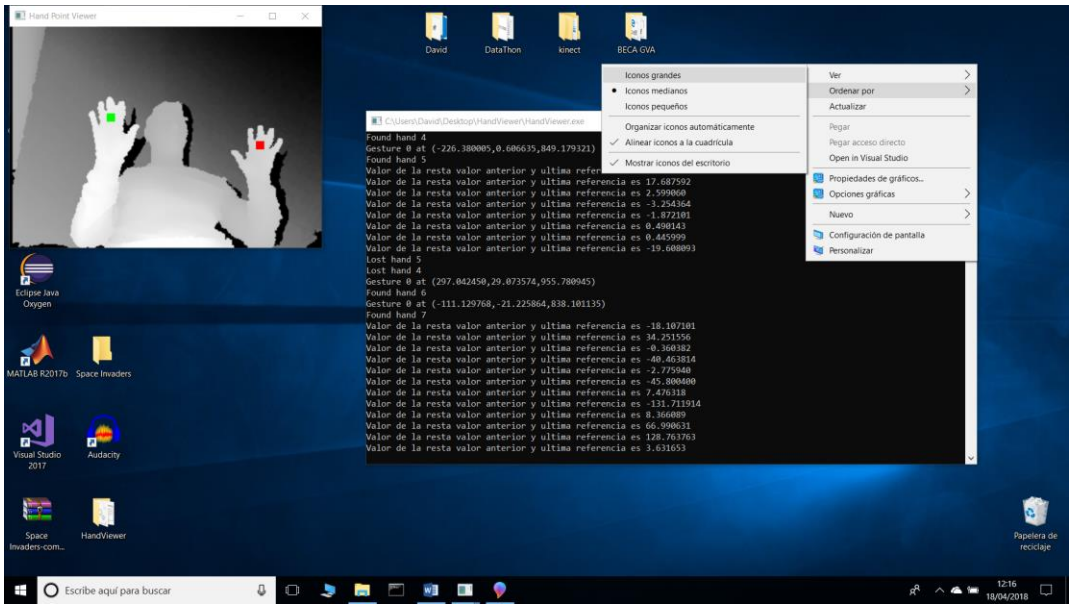
[8] Detección de una mano

2º. Hacer ‘Click’.

Una vez detectada la mano, el cursor hará un ‘click’ en el momento que se detecte poco movimiento de esta, es decir, cuando este quieta.

3º. Hacer ‘Click derecho’.

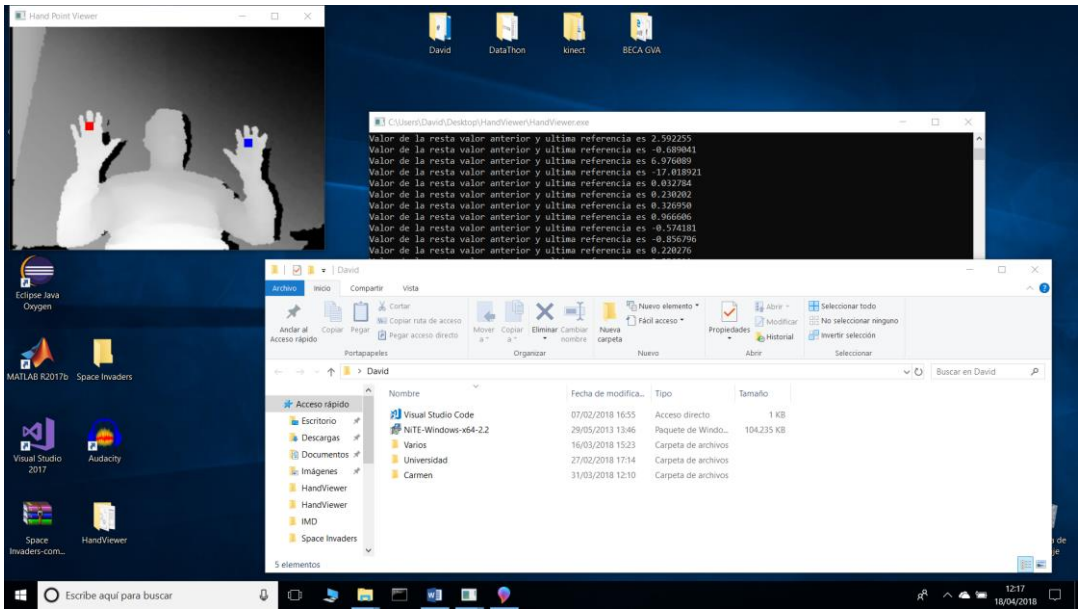
Para llevarlo a cabo, será necesario que antes haya detectado la segunda mano. Para hacer ‘click derecho’ bastará con mover bruscamente a un lado con la segunda mano detectada. En la imagen[9], no se puede apreciar el movimiento necesario que ha permitido hacer el *‘click derecho’*.



[9] Realización ‘click derecho’

4º. Hacer ‘Doble Click’.

Para ello, al igual que en el tercer apartado, tendrás que tener la segunda mano detectada. En este caso, se necesitará hacer la animación de pulsar con la mano mirando a la cámara. En la imagen[10] mostrada no se puede apreciar el movimento necesario para realizar el ‘*dobleclick’.*



[10] Realización ‘*doble click’*

Nota: Detectar más de dos manos, no causará ningún efecto. Sólo se estará utilizando las dos primeras detectadas.

7. ASPECTOS A MEJORAR

Actualmente disponemos de una ventana en el que nos vemos a nosotros y un terminal para ver si estamos bien colocados y si nos está detectando la mano (útil como interfaz de desarrollo y depuración). Sería conveniente poder mover el cursor en pantalla sin crear dichas ventanas y crear algún pequeño icono para ser consciente de que la aplicación se esta ejecutando correctamente y así no interferir al usuario en el uso de la aplicación con ventanas "molestas" que no puede quitar porque si no se cierra la aplicación.

Por otro lado, sería conveniente poder realizar una detección mas minuciosa de la mano, para poder asignar punteros por ejemplo a cada dedo y según una serie de gestos poder realizar acciones sobre el ordenador, eliminando así la necesidad de una segunda mano.

Por último, podría añadirse a este programa la capacidad para poder arrastrar imágenes, poder ampliarlas o alejarlas o algún tipo de acceso rápido según la acción realizada.

8. BIBLIOGRAFÍA

-[4] Mouse Event para Windows en: <https://msdn.microsoft.com/en-us/library/windows/desktop/ms646260(v=vs.85).aspx>

-Enlace descarga Nite2.2: <https://bitbucket.org/kaorun55/openni-2.2/src/2f54272802bf/NITE%202.2%20%CE%B1/?at=master>

-Enlace descarga OpenNi2: <https://structure.io/openni>

-PrimeSense 3D Sensors. Disponible en <http://www.i3du.gr/pdf/primesense.pdf>

-PrimeSense documentación en: <http://www.openni.ru/wp-content/uploads/2013/02/NITE-Algorithms.pdf>

-ASUS XTION PRO LIVE web en: <https://www.asus.com/us/3D-Sensor/Xtion_PRO_LIVE/>