Universidad de Santiago de Chile
Facultad de Ingeniería
Departamento de Ingeniería Mecánica.
Automatización y Robótica.

Proyecto Robótica
Informe Avance Proyecto Nº 2

Profesor: Michael Miranda.

Alumnos: Iván Fernández

Claudio Canales D.

Índice

Objetivo 3
Metas específicas 3
Procedimiento 3
Introducción 4
Desarrollo 5
Conclusión 15
Bibliográfica 15

Objetivo

- Encontrar un entorno de desarrollo interactivo de código abierto que permita la manipulación de los datos recolectados del Kinect v1
- Aprender a utilizar y modificar el lenguaje de programación del entorno interactivo encontrado

Metas específicas

- Instalar y aprender a utilizar el software libre Processing.
- Conocer los códigos para visualizar la cámara RGB en Processing.
- Conocer los códigos para visualizar el sensor infrarrojo de distancias en Processing.
- Interiorizar en los posibles usos de Kinect en Processing.
- Hallar librerías que obtengan el esqueleto de un individuo partir de algoritmos de alto nivel en Processing.
- Conectar a Matlab el Kinect v1 y evaluar si es óptimo para obtener la esqueletizacion de objetos.

Procedimiento

- De las páginas oficiales de los programas se descargar versiones coincidentes de OpenNI, NITE y Processing hasta encontrar una versión que sea compatible con la computadora.
- La versión compatible de los programas debe ser la correcta para que la librería SimpleOpenNI (Processing: Esqueletizacion) funciones correctamente, ya que los códigos interiores de esta librería son muy sensible a cambios de versión y de bites (32 o 64 bits)
- Una vez tener instalados correctamente los programas y la librería SimpleOpenNI, se procede a ver los ejemplos que trae esta librería y la del programa OpenNI. Esta trae consigo códigos que nos ayudan a entender la plataforma y los algoritmos para obtener la imagen de colores y distancia del Kinect v1.
- Después se busca un ejemplo de un código en la página Github para obtener los puntos de la esqueletizacion de un individuo en Processing por medio de la librería SimpleOpenNI. Se analiza y se comprende el codigo.
- Finalmente se compara él envió de datos de la esqueletizacion entre el programa Processing y Matlab donde se elige cual es mejor para nuestro proyecto.

Introducción

OpenNI

Tras el gran interés de los estudiantes que utilizan Kinect y la investigación de programas de código abierto, la empresa PrimeSense lanzó su propio software para trabajar con Kinect para que los programadores puedan acceder a la información de Kinect. Este se llama OpenNI (Open Natural Interaction) y es un SDK de código abierto utilizado para el desarrollo de bibliotecas y aplicaciones de middleware de detección 3D principalmente. Cabe destacar que el drive SDK de Microsoft proporciona desarrolladores que trabajan solamente en Windows, por ende OpenNI es la mejor opción para quien quiera trabajar en cualquier plataforma y lenguaje de programación.

NITE

Con el sistema OpenNI el programador puede acceder a los datos de profundidad y cámara de colores de Kinect. OpenNI está bajo una licencia. Sin embargo, una de las características más interesantes de OpenNI es el seguimiento de los usuarios que no está cubierta por la licencia. En su lugar esta proporcionada por un módulo externo llamado NITE. NITE no está disponible bajo una licencia de código abierto. Se trata de un producto comercial que pertenece a PrimeSense. Su código fuente no está disponible pero PrimeSense proporciona una licencia gratuita que puede utilizarse para hacer proyectos que utilizan NITE con OpenNI. Es preciso señalar que algoritmos utilizan la profundidad, el color, el IR y la información de audio recibidos del dispositivo de hardware, lo que les permite realizar funciones tales como la localización y el seguimiento a mano, un analizador de escenas (separación de usuarios de fondo), seguimiento del esqueleto del usuario exacto, reconocimiento de varios gestos y más.

Processing

Processing es una plataforma informática, de lenguaje de programación basado en Java que, entre una de todas las cosas que se puede utilizar, facilita procesos visuales con poco código y la obtención de datos en tiempo real. Mediante Processing podemos dibujar gráficos en dos y tres dimensiones facilitando la manipulación de los datos recolectados de Kinect v1.

El entorno de desarrollo de Processing (PDE) es sencillo y fácil de usar. Los programas se escriben en el editor de texto y se ejecutan pulsando start. Cada programa se escribe en un sketch que por defecto es una subclase de Papplet de Java que implementa la mayor parte de las funciones de Processing. También permite crear clases propias en el sketch pudiendo así usar tipos de datos complejos con argumentos y evita la limitación de utilizar exclusivamente datos tales como enteros, caracteres, números reales y color. Los sketchs se almacenan en sketchbook que no es más que un directorio dentro del ordenador.

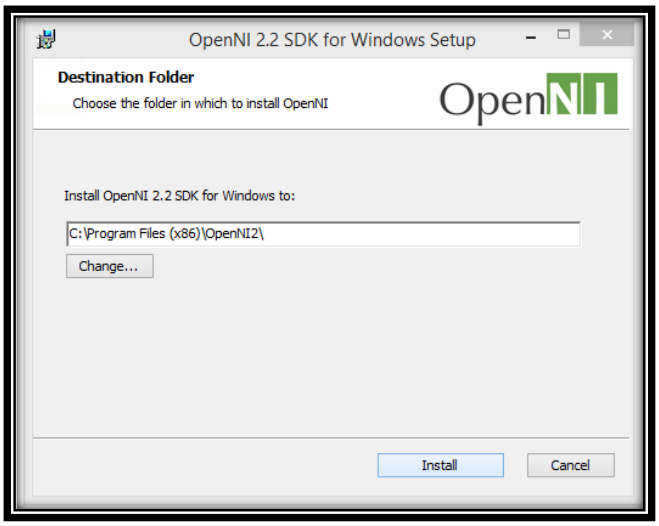
Es posible ampliar las capacidades de Processing mediante el uso de extensiones y bibliotecas. Para utilizar Kinect en Processing se importan las librerías *"OpenCV for Processing"* (librería software open-source de visión artificial y machine learning con algoritmos permiten identificar objetos, caras, clasificar acciones humanas en vídeo, etc.) *"Open Kinect for Processing"* (Para usar el sensor de Microsoft Kinect en Processing y permite acceder a los datos de cámara RGB y de profundidad) y *"SimpleOpenNI"* (nos proporciona las capacidades de OpenNI en Processing, como por ejemplo la esqueletizacion e interfaz gráfica 2D y 3d de alto nivel).

Desarrollo

Parte 1: Instalación de entornos de desarrollo

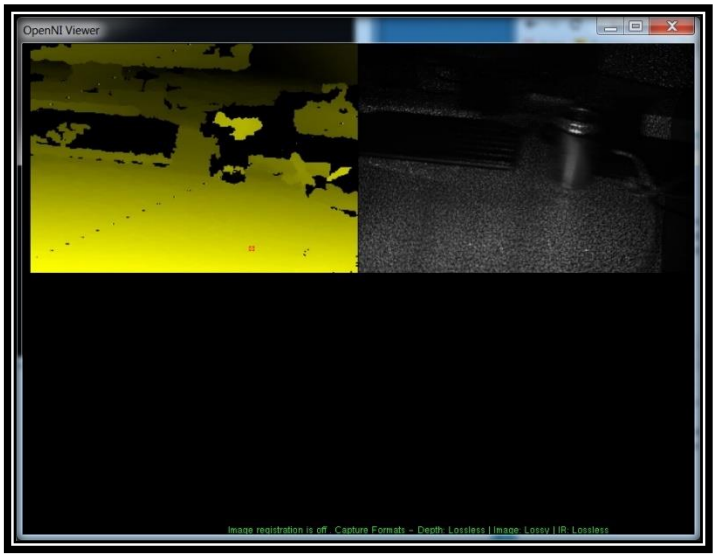
OpenNI

Para descargar OpenNI SDK se accede a la página oficial http://openni.ru/openni-sdk/index.html y se descarga la versión para Windows *"OpenNI 2.2.0.33 Beta (x86) (32 bits)"*. Se abre el archivo descargado y se siguen las instrucciones que indica el setup:



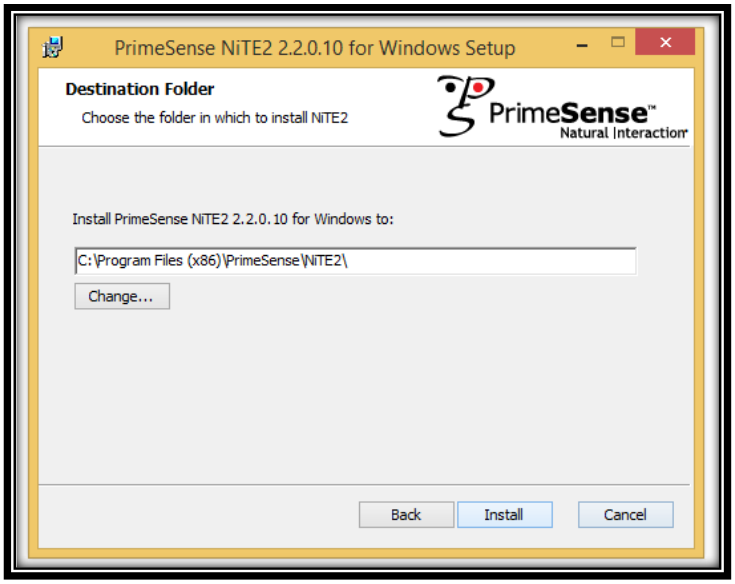
Nota: El código fuente OpenNI está disponible en GitHub

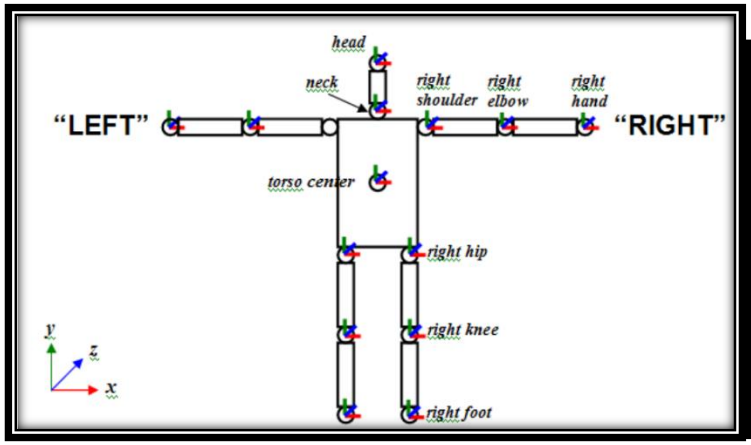
Para verificar la instalación correcta, se abre Viewer OpenNI (aplicación de OpenNI) con el Kinect conectado. Debe aparecer en la interfaz la visualización de la cámara infrarroja o RGB.



NITE

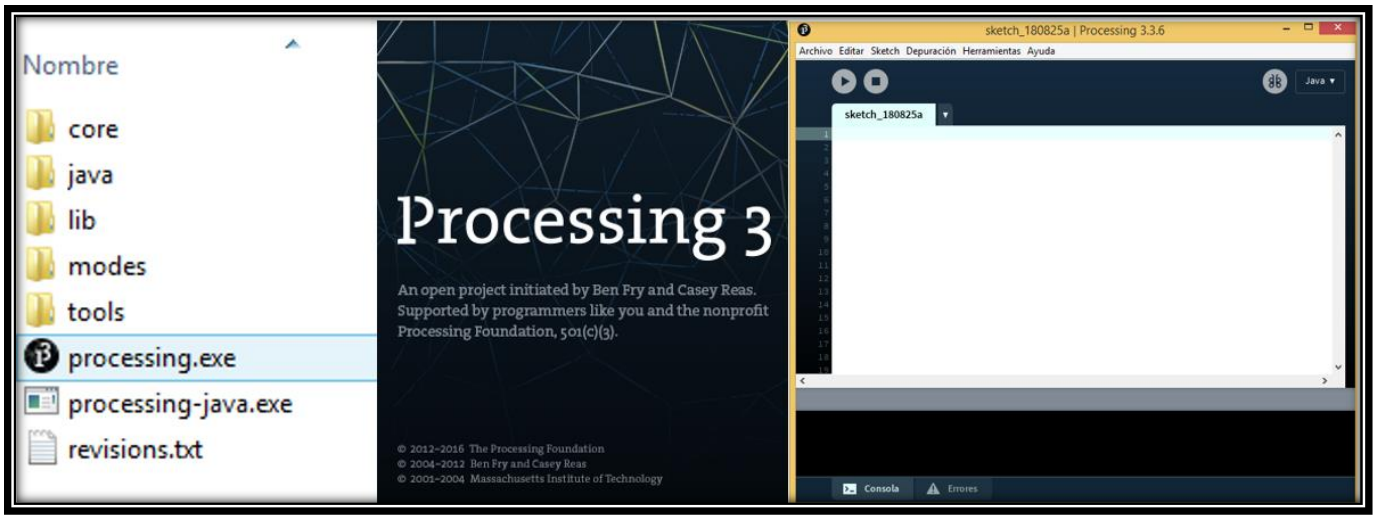
El middleware 3D PrimeSense NITE también se descarga de la página oficial http://openni.ru/files/nite/index.html donde se descarga la versión "*NITE 2.2.0.11".* El único prerrequisito que pide es tener instalado OpenNI 2.2. Se aceptan las condiciones de termino y se apretar el botón instalar.





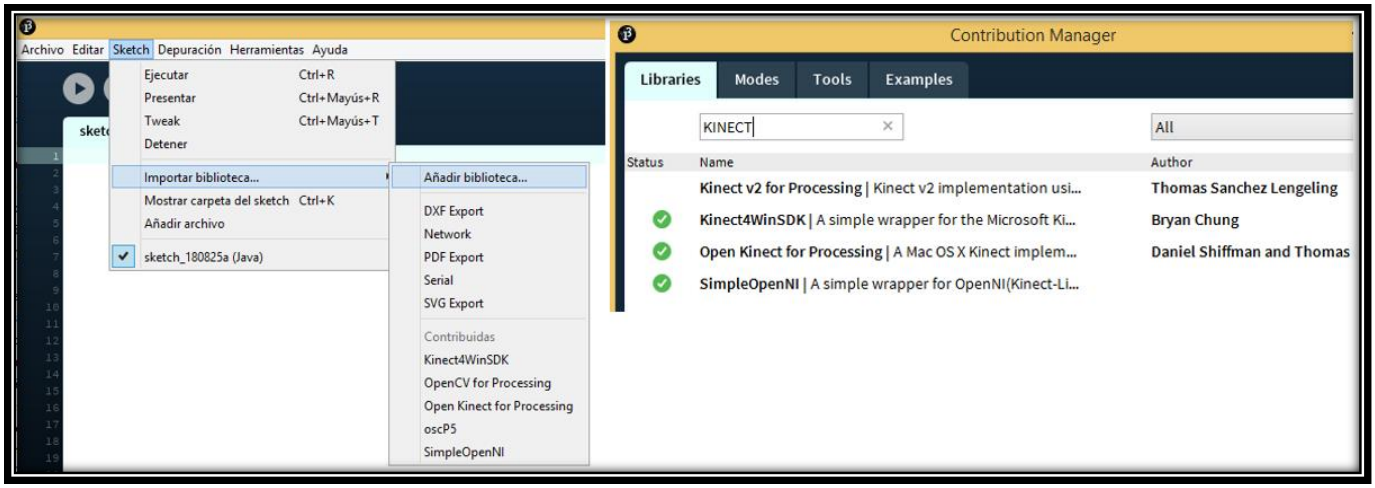
Processing

Para instalar Processing en la computadora hay que ingresar a la página oficial https://processing.org/. Se descargar la versión con sistema operativo Windows de 32 bits: "*Processing - 3.3.6 – Windows 32".* Para abrir Processing solo se debe abrir el archivo *processing.exe* de la carpeta comprimida descargada anteriormente.



Librerías Processing

Para buscar e importar librerías, desde la ventada de Processing se hace clic en *Sketch > Importar librerías > Añadir a la biblioteca*. Esto abrirá la ventana *Contribution Manager* donde se podrá buscar todo tipo de librerías de diferentes autores gratuitamente. Otra forma de importar una librerías es descargarla directamente de la red y el archivo de la librería copiarlo dentro de la carpeta de Processing llamada *"Libraries".* Para verificar una importación correcta se debe observar en *Sketch > Importar librerías* que abajo este el nombre de la librería requerida.



Las librerías que se importaran para el reconocimiento de un esqueleto y el envió de los datos a cualquier aplicación son:

- OpenCV for Processing: con algoritmos permiten identificar objetos, caras, etc.
- Open Kinect for Processing: Para usar el sensor de Microsoft Kinect en Processing
- OscP5: Implementación Open Sound Control para la transmisión de datos.
- SimpleOpenNI: OpenNI en Processing

Nota: GitHub es una plataforma de desarrollo inspirada en la forma en que trabajas. Desde el código abierto hasta el negocio, puede alojar y revisar el código, administrar proyectos y crear software junto con 30 millones de desarrolladores. De esta página se debe descargar librería *SimpleOpenNI* y para encontrarla en el buscador se escribe: *simple-openni-master* o solo se ingresa a la página: https://github.com/wexstorm/simple-openni

Se pueden encontrar más versiones y archivos compatibles con sistemas operativos de la librería *SimpleOpenNI* de la página https://code.google.com/archive/p/simple-openni/downloads ya que está en constante mejora.

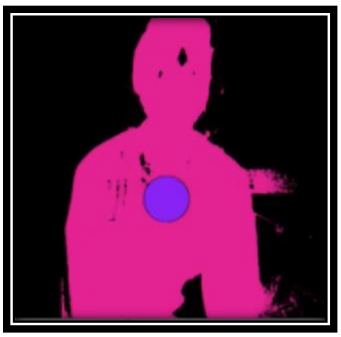
Parte 2: Códigos para obtener la cámara RGB y distancias a partir de sensor infrarrojo en Processing

sketch_1

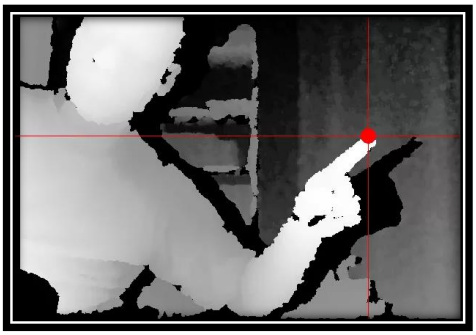
```
//Lo primero es incluir las declaraciones de importación adecuadas en la parte superior de su código:
import org.openkinect.processing.*;

//Además de una referencia a un objeto Kinect, es decir
Kinect kinect;

//Luego setup(), puedes inicializar ese objeto kinect:
void setup() {
  kinect = new Kinect(this);
  kinect.initDevice();
  kinect.initDepth();
}

//Acceder a los datos del sensor de kinect
//Actualmente, la biblioteca pone los datos a su disposición de cinco maneras:
//PImage (RGB) de la cámara de video kinect.
//PImage (grayscale) de la cámara IR de Kinect.
//PImage (grayscale) con el brillo de cada píxel asignado a la profundidad (más brillante = más cerca).
//PImage (RGB) con el matiz de cada píxel asignado a la profundidad.
//int[] array con datos de profundidad sin formato (números de 11 bits entre 0 y 2048).


//Dibujar RGB y profundidad
void drag(){
  // Tamaño de la ventana en pixeles para la visualizacion de RGB o profundidad
  size(512,424)

  //Utilizar el Kinect como una cámara web vieja y normal
  //Con kinect v1 no se puede obtener tanto la imagen de video como la imagen de IR. Ambos son devueltos a través de getVideoImage ()
  PImage img = kinect.getVideoImage();
  image(img, 0, 0);

  //Imagen de profundidad en escala de grises:
  PImage img = kinect.getDepthImage();
  image(img, 0, 0);
}

//Datos de profundidad sin procesar:
//Los valores de profundidad sin procesar oscilan entre 0 y 2048
int[] depth = kinect.getRawDepth();

//Imagen de profundidad de color
kinect.enableColorDepth(true);

//FUNCIONES UTILES
//initDevice() - comienza todo (video, profundidad, IR)
//activateDevice(int) - activar un dispositivo específico cuando se conectan varios dispositivos
//initVideo() - solo iniciar video
//enableIR(boolean) - encender o apagar la imagen de la cámara IR (v1 solamente)
//initDepth() - solo comienza la profundidad
//enableColorDepth(boolean) - activar o desactivar los valores de profundidad como imagen en color
//enableMirror(boolean) - duplicar la imagen y los datos de profundidad
//PImage getVideoImage() - tomar la imagen de video RGB
//PImage getDepthImage() - agarra la imagen del mapa de profundidad
//int[] getRawDepth() - agarrar los datos de profundidad sin procesar
//float getTilt() - obtener el ángulo del sensor actual (entre 0 y 30 grados)
//setTilt(float) - ajustar el ángulo del sensor (entre 0 y 30 grados)
```

Parte 3: Posibles usos del Kinect en Processing

- Rastreo promedio de puntos (Centroide)

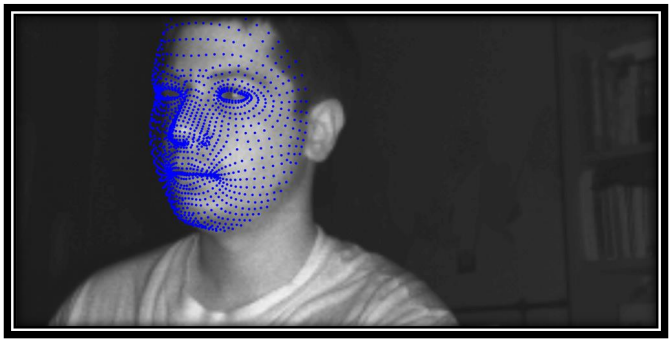


- Detección punto más cercano, más alto, más bajo, más alejado



- Detección de bordes

- Mapeo 3D de superficies

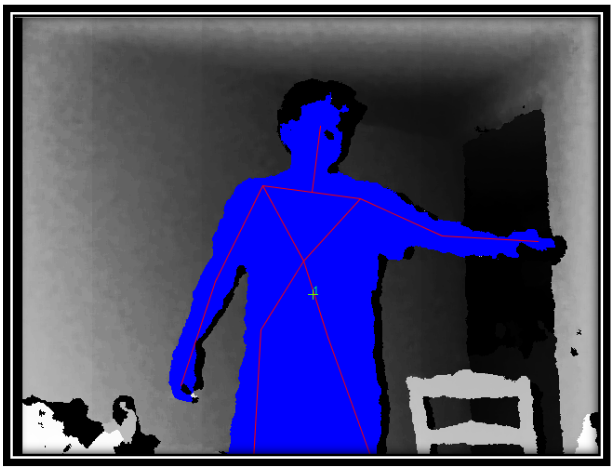


Parte 4: Esqueletizacion en Processing (SimpleOpenNI)

Se obtiene el modelo del esqueleto a través de la utilización de la librería SimpleOpenNI. Para una descripción más detallada del código se encuentra información en el libro *"Making.Things.See - Greg.Borenstein".*

```
1.  //Se importa la libreria SimpleOpenNI
2.  import SimpleOpenNI.*;
3.  //Se inicializa la instancia context.
4.  SimpleOpenNI  context;
5.  //se utiliza este Array para los  colores de las personas detectadas.
6.  color[]       userClr = new color[]{ color(255,0,0),
7.                                       color(0,255,0),
8.                                       color(0,0,255),
9.                                       color(255,255,0),
10.                                      color(255,0,255),
11.                                      color(0,255,255)
12.                                    };
13. PVector com = new PVector();
14.
15. PVector com2d = new PVector();
16.
17. void setup()
18. {
19.   //Se inicializa una ventana de 640*480 donde aparecerá el esqueleto
20.   size(640,480);
21.   //Se crea la instancia context
22.   context = new SimpleOpenNI(this);
23.   if(context.isInit() == false)
24.   {
25.      println("No se puede inicializar SimpleopenAI, no esta conectada la camara");
26.      exit();
27.      return;
28.   }
29.   // Permite la obtención de la profundidad
30.   context.enableDepth();
31.   // Crea el esqueleto para todas las uniones
32.   context.enableUser();
33.  //Fondo de la pantalla
34.   background(200,0,0);
35.   stroke(0,0,255);
36.   strokeWeight(3);
37.   smooth();
38. }
39.
40. void draw()
41. {
42.   // Actualiza la cámara
43.   context.update();
44.
45.   //Dibuja la imagen de profundidad
46.
47.   //image(context.depthImage(),0,0);
48.
49.   //Presenta la imagen RGB en la ventana, en la posición (0,0)
50.
51.   image(context.userImage(),0,0);
52.
```

```
53.
54.
55.  // Dibuja el esqueleto si esa disponible
56.
57.  //Declara un Array de los usuarios en la pantalla
58.   int[] userList = context.getUsers();
59.   for(int i=0;i<userList.length;i++)
60.   {
61.     //Prueba para saber si está realizando Esqueletizacion
62.
63.     if(context.isTrackingSkeleton(userList[i]))
64.     {
65.       stroke(userClr[ (userList[i] - 1) % userClr.length ] );
66.
67.       //LLAMA A LA LA FUNCION DRAW , QUE ESTA AL INFERIOR
68.
69.       drawSkeleton(userList[i]);
70.     }
71.     // Dibuja el centro de masa
72.
73.     if(context.getCoM(userList[i],com))
74.     {
75.       context.convertRealWorldToProjective(com,com2d);
76.       stroke(100,255,0);
77.       strokeWeight(1);
78.       beginShape(LINES);
79.         vertex(com2d.x,com2d.y - 5);
80.         vertex(com2d.x,com2d.y + 5);
81.
82.         vertex(com2d.x - 5,com2d.y);
83.         vertex(com2d.x + 5,com2d.y);
84.       endShape();
85.
86.       fill(0,255,100);
87.       text(Integer.toString(userList[i]),com2d.x,com2d.y);
88.     }
89.   }
90. }
91.
92.       // Dibuja el esqueleto con las uniones seleccionadas
```

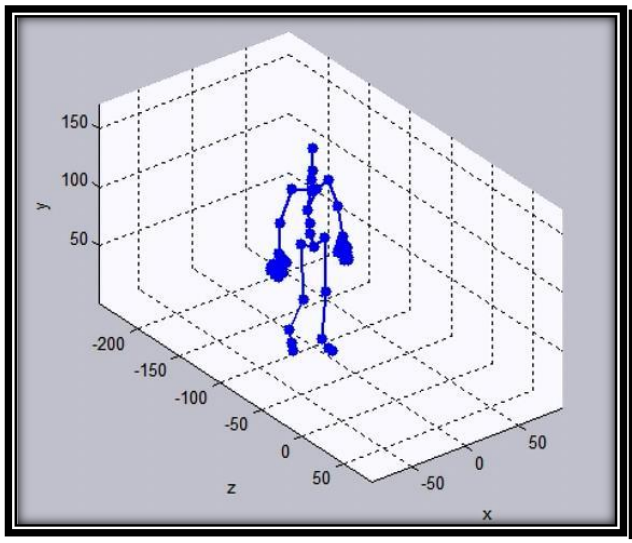


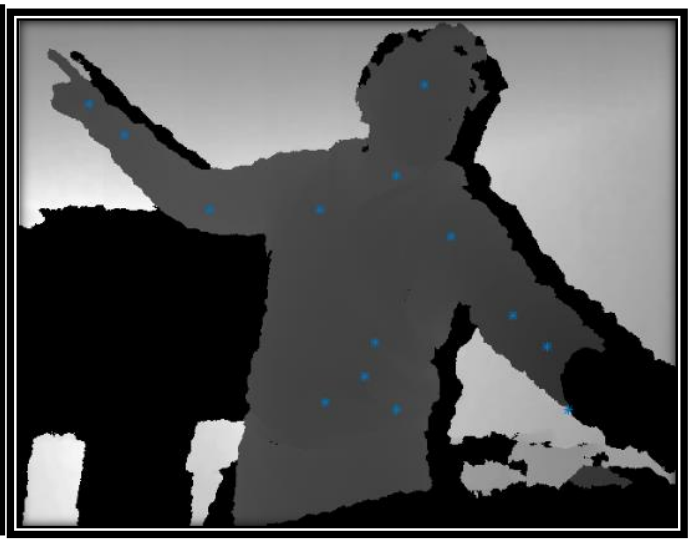
Parte 5: Conectar Kinect en Matlab y obtener esqueleto

Para utilizar Kinect en el programa Matlab se debe descargar un paquete llamado *"Image Acquisition Toolbox"* que permite adquirir datos del sensor de imagen directamente en MATLAB y Simulink. El código para visualizar el esqueleto es el siguiente:

```
clc; clear all ; close all
% INICIALIZACION DE KINECT, CAMARA RGB Y SENSOR DE DISTANCIA
imaqreset;
depthVid= videoinput('kinect',2);
triggerconfig (depthVid, 'manual');
depthVid.FramesPerTrigger=1;
depthVid.TriggerRepeat=inf;
set(getselectedsource(depthVid),'TrackingMode','Skeleton');
viewer=vision.DeployableVideoPlayer();
start(depthVid);
himg=figure;
pause(5)
% SELECCIONAR PUNTOS SELECCIONADOS Y LOS DIBUJA
while ishandle(himg)

    trigger(depthVid);
    [depthMap,~,depthMetaData]=getdata(depthVid);
    %[imgColor, ts_color, metaData_Color] = getdata(vid2);
    %imshow(depthMap,[0 4096]);
    %imshow(vid2)
    %size(depthMetaData)
    if sum(depthMetaData.IsSkeletonTracked)>0
        skeletonJoints = depthMetaData.JointDepthIndices(:,:,depthMetaData.IsSkeletonTracked);
        %hold on;
        %plot(skeletonJoints(:,1), skeletonJoints(:,2),'*');
        %for ii=1:20
        %    t(ii)=depthMap(skeletonJoints(ii,1), skeletonJoints(ii,2))
        %end
        %plot3(t,skeletonJoints(:,1), skeletonJoints(:,2),'*');
        %cont=0;

        % SELECCIONA PUNTOS DEL ESQUELETO DENTRO DE 640X480
        iii=0;
        for ii=1:20
            n=skeletonJoints(ii,1);
            m=skeletonJoints(ii,2);
            if (((n<=0)||(m<=0))||((n>640)||(m>480)))
                %cont=1
                %print ('algo malo punto')
                %pause(1)
            else
                %cont=2
```

```
                iii=iii+1;
                nn(iii)=skeletonJoints(ii,1);
                mm(iii)=(skeletonJoints(ii,2));
                mmm(iii)=floor(-(skeletonJoints(ii,2))+481) ;
                t(iii)=depthMap(mm(iii),nn(iii));
            end
        end
        %DA VUELTA LA IMAGEN O PUNTOS
        cont=0
        for ii=1:iii
            if ((abs(t(3)-t(ii)))<1000)
               cont=cont+1
               nnx(cont)=nn(ii);
               mmx(cont)=mm(ii);
               mmmx(cont)=mmm(ii) ;
               tx(cont)=depthMap(mmx(ii),nnx(ii));
            end
        end

        % DIBUJAR PUNTOS DE DISTANCIA SELECCIONADOS
        plot3(tx,nnx,mmmx,'*');
        hold on
        plot3(t(3),nn(3),mmm(3),'r*');
        hold off
        axis([0,4096,0,640,0,480]);
        grid on
    end
end
```





Conclusión

La conexión del dispositivo al computador fue realizada con éxito, sin embargo existieron inconvenientes ya que los drivers de Microsoft para Kinect ahora son limitados para el uso en programas abiertos, en vista que Windows compro la marca "Kinect Xbox 360" y empezaron a vender su propio modelo "Kinect for Windows". En el mismo plano, los drivers PrimeSensor fueron vendidos a la empresa Apple, dificultando la instalación actualmente en los ordenadores de gente común y corriente. Cabe decir que los programas instalados deben ser de la misma versión y de 32 bits para que el Kinect v1 pueda utilizarse, dificultando notoriamente la instalación.

Para procesar la información extraída del Kinect y visualizarla en animaciones 2d o 3d se necesita una tarjeta de video apropiada y con los drivers actualizados, aumentando los costos de cualquier proyecto de kinect que tenga este objetivo.

Se puede apreciar de las imágenes obtenidas, que el dispositivo Kinect funciona de forma correcta y puede graficar las imágenes de profundidad y RGB de forma correcta, sin visualizar ningún tipo de retraso o problema alguno, esto es debido a la correcta implementación de las librerías.

El proceso de esqueletización presenta una velocidad de respuesta alta, ya que el código se ejecuta de forma continua y presenta una mínima latencia (5ms).

Se puede observar, que la esqueletización se ejecuta de forma exitosa y que el modelo funciona bien, independiente de las características físicas del individuo, ropa, tamaño, entre otros.

El algoritmo utilizado para la esqueletización, que se basa en "Random Forest", funciona según lo esperado, representando una opción fiable, rápida y económica. Incluso con varios usuarios, el proceso se lleva a cabo sin mayores problemas.

El proceso de comunicación a través de los protocolos OSC y UDP, funcionan exitosamente. Aun así, falta desarrollo del código debido a que fueron probados con el mismo programa y lo ideal es que la comunicación se utilizado entre dos programas diferentes, para realizar la animación.

Bibliográfica

- Librería SimpleOpenNI*: "Making Things See* "– Greg Borenstein
- Códigos de informáticos en Processing: https://github.com/
- Paquete para Kinect v1 y esqueletizacion en Matlab: https://www.mathworks.com/help/imaq/acquiring-image-and-skeletal-data-using-the-kinect.html