Multimodal and natural computer interaction

E. Sampedro Sánchez

Tutora: Martina Eckert

ETSIST Universidad Politécnica de Madrid
Madrid, España

Abstract

Es necesario un entendimiento de que es la realidad aumentada, como está presente en nuestras vidas y como podría llegar a mejorarlas, estudiando los medios en los que se puede conseguir, con que dispositivos y en que ámbitos, pero teniendo presente que la interacción hombre-máquina debe ser lo más natural posible es necesario encontrar métodos que no sean intrusivos, por lo que no hay nada menos intrusivo que no llevar nada, simplemente que unos sensores recojan información y traten esa información para hacer la vida más fácil; estos sensores nos los aporta la Kinect.

Keywords: interfaces naturales, detección, realidad aumentada, kinect

I. INTRODUCCIÓN

En el mundo de las telecomunicaciones el concepto de Internet of Things ha empezado a ganar importancia debido a que los distintos objetos y personas a nuestro alrededor necesitan estar conectados para interactuar entre ellos y proporcionar una eficiencia a los nuevos sistemas.

Pero en una sociedad en la que cada vez están más presentes las máquinas, es necesario que no se olvide el factor humano, la naturalidad con la que los humanos se comunican hay que saber exportarla a un mundo que hace años podría parecernos ciencia ficción y ahora se nos muestra como un futuro no muy lejano.

II. ESTADO DEL ARTE

En este apartado se ofrece una visión de distintas técnicas de detección, que son gestos, corporal y facial, realizando una comparación entre los trabajos realizados más interesantes encontrados acerca de ello (Anexo 1), centrándome en sus tecnologías, autores y años de publicación. Después se descubrió que el denominador común de muchas de estas técnicas era la Kinect, el sistema de reconocimiento del movimiento desarrollado por Microsoft.

1. Técnicas de detección

1.1. Gestos

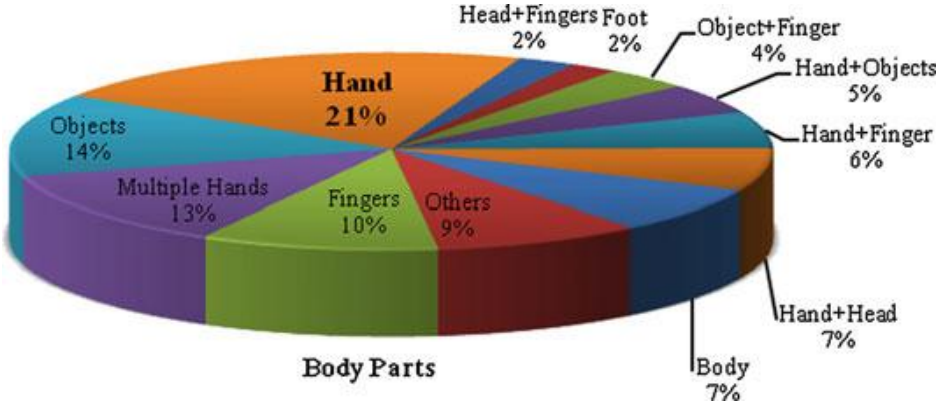

Figura 1. Muestra las diferentes partes del cuerpo u objetos identificados en [1] empleados para gestos

Primero se especifican cuales son las técnicas de detección de los gestos, se pueden observar las técnicas más concretas en [1]:

- Color: Se han propuesto varios espacios de color incluyendo RGB, RGB normalizado, HSV, YCrCb, YUV, etc. En general, la segmentación de color hace que se pueda confundir las manos con los objetos de fondo que tengan un color similar al color ce la piel, por lo que se utiliza la substracción de fondo
- Forma: Se utiliza la forma característica de las manos; si se detecta correctamente, el contorno representa la forma de la mano y por lo tanto no es directamente dependiente de punto de vista, color de la piel y la iluminación, pero puede llegar a tener puntos de vista degenerados, por lo que este método se combina con la detección del color de piel y eliminación de los objetos de fondo
- Valores de pixeles: Detección de las manos en imágenes de nivel de gris en base a su apariencia y textura.
- Modelo 3D: Una de las ventajas de estos métodos es que se puede lograr la detección independiente de la vista. Los modelos 3D empleados deben tener suficientes grados de libertad para adaptarse a las dimensiones de la mano (s) presentes en una imagen.
- Movimiento: Asume que el único movimiento en la imagen es debido al movimiento de la mano. En los enfoques más recientes, la información de los movimientos se combina con señales visuales adicionales. En el caso de cámaras estáticas, el problema de la estimación de movimiento se reduce a la de fondo de mantenimiento y posterior sustracción.

Las aplicaciones de estas técnicas son muy amplias, hoy en día hasta las televisiones poseen la tecnología para interactuar con el usuario mediante gestos, pero las primeras en aparecer fueron las aplicaciones controladas solo por un dedo en lugar de un ratón, que fueron estas:

- "Finger-pointer": Pointing interface by image processing. (1994): interfaz genérica que estima la localización y orientación del ”pointing finger”
- “FingerPaint”, Finger tracking as an input device for augmented reality (1995): el movimiento del dedo índice del usuario indica la línea de dibujo; el puntero se determina mediante la búsqueda de la posición de la imagen en la que se minimiza la suma de cuadrados de las diferencias con una plantilla de referencia.
- A Usable Real-Time 3D Hand Tracker (1995): Se usan algoritmos para extraer la posición y la orientación plana de la mano, y los ángulos de las articulaciones de los dedos.
- “FingerMouse”, Unencumbered gestural interaction (1996): Movimientos de los dedos en 2D se

interpretan como el movimiento del ratón del ordenador.

- Virtual three-dimensional blackboard: Three-dimensional finger tracking with a single camera (2004): seguimiento de un dedo humano a partir de una secuencia de monocular de imágenes para implementar una aplicación de pizarra 3D; para recuperar la tercera dimensión se utiliza el hecho de que el movimiento del brazo humano es muy limitado.

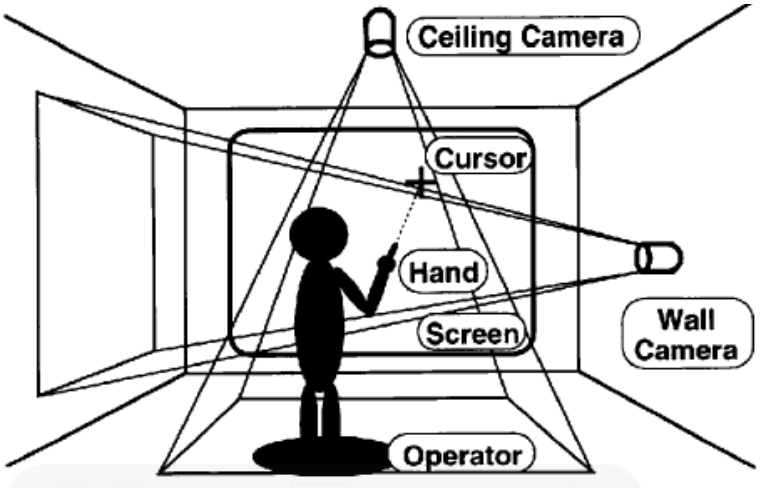

Figura 2. Finger-pointer

1.2. Facial

Reconocer emociones o variaciones naturales en el rostro puede permitir aplicar estos datos en el campo de la realidad aumentada, como puede ser cambiar un entorno para amoldarlo al estado en el que se encuentra el usuario. La identificación de emociones en el rostro recaba información asociada a las formas de ojos, nariz, boca, ubicación de vértices, arrugas, surcos y protuberancias. Estimaciones, como la edad, también podrían inferirse de los métodos empleados para detectar emociones si la atención se centra en surcos o brillo de la piel.

La detección del rostro es el primer paso de los algoritmos de reconocimiento de cara y por lo tanto es de crucial importancia que su desempeño sea lo mejor posible. Aquí se decide si la imagen o video incluye caras; si las incluye, se identifican sus posiciones y se segmentan separándolas del fondo de la imagen. Los factores principales que se deben tener en consideración en la detección de rostros son: luminosidad, orientación, escala, textura y accesorios u objetos que puedan afectar a la geometría esperada (gafas, bufandas, etc.); los métodos más comunes para realizar esta detección son:

- Plantillas: se calculan valores de correlación entre una imagen de entrada y sus plantillas. Si los valores de correlación alcanzan un determinado umbral se acepta que la imagen contiene una cara. Los valores de correlación se calculan por separado para los contornos del rostro, ojos, boca y nariz.
- Movimiento: usa un cierto número de imágenes consecutivas, algunas veces dos o tres en secuencia. En este planteamiento comparan frames como en un análisis de imágenes estáticas y el movimiento se determina buscando correspondencias entre pares de puntos de interés en la secuencia.
- Tonalidad: Los mapas de reflectancia capturan la dependencia del brillo con la orientación de la superficie, existe una correspondencia única de la orientación de la superficie a la reflectancia especificada por el mapa de reflectancia. Sin embargo la correspondencia inversa no es única, un número infinito de orientaciones de superficies producen el mismo brillo, por lo tanto, un contorno constante conecta dicho conjunto de orientaciones en el mapa de reflectancia
- Texturas: emplean parámetros característicos de una matriz de dependencias de niveles de gris entre celdas de píxeles; el modelo de texturas faciales busca caras sobre imágenes en color detectando tonos

naranja. La ventaja de este método es que es capaz de localizar rostros que no tengan una vista frontal y es tolerante a oclusión por barba o gafas.

- Visión estereoscópica: Para extraer la correspondencia de las imágenes de un sistema estereoscópico existen dostécnicas:
    - Por áreas: utilizan la correlación cruzada entre patrones de intensidad en la vecindad local de un píxel en una imagen con patrones también de intensidad en una vecindad correspondiente de un píxel en la otra imagen del par estereoscópico.
    - Por características: utilizan representaciones simbólicas obtenidas de las imágenes de intensidad en lugar de las intensidades directamente. Las características utilizadas normalmente son: puntos de borde aislados, cadenas de puntos de bordes y regiones delimitadas por bordes.
- Conocimiento humano: se centra en las características más notorias de un rostro, intenta traducir el conocimiento humano sobre detección en reglas estrictas. Por ejemplo, es común que una cara contenida en una imagen aparezca con dos ojos separados simétricamente, una nariz y una boca; así entonces, las relaciones entre estos rasgos pueden representarse mediante distancias relativas.
- Características invariantes: tratan de reproducir la habilidad de los seres humanos para detectar caras y objetos en diferentes posturas y condiciones del ambiente empleando un mínimo esfuerzo.
- Apariencias: uso de técnicas estadísticas.
- Modelos geométricos: comparan un modelo de puntos con una nueva imagen usando variante del algoritmo de maximización de la esperanza.
- Modelos 3D: Manejan variaciones de iluminación y expresiones faciales y pueden dividirse en dos categorías: métodos 3D simples que usan sólo representaciones de la superficie o forma del rostro, y métodos que usan formas 3D e imágenes 2D
- Infrarrojos: La ventaja principal de estas técnicas sobre los sensores visibles, es que las imágenes IR son independientes de la iluminación ambiental, la luz IR solamente se emite y no se refleja

1.3. Corporal

Las técnicas de seguimiento de cuerpos, o body-tracking, despiertan gran interés, sobre todo por su aplicación en seguridad y salud. Sus inicios se remontan a la detección simple de objetos por visión artificial y se ha abordado frecuentemente con algoritmos bayesianos.

Es necesario investigar la posibilidad de crear sistemas que sean capaces de reconocer las emociones de forma independiente de la acción que la persona está haciendo. Esto es muy importante dado el alto grado de libertad del cuerpo y dado que estos sistemas se puede implementar de forma ubicua.

2. Realidad Aumentada

Visión directa o indirecta en tiempo real de un entorno del mundo real físico que ha sido mejorado/aumentado por la adición de información virtual generada por ordenador. En la realidad virtual los dispositivos pueden ser clasificaos en tres categorías en función de la posición entre el espectador y el mundo real:

- Head-worn: video/optical see-through head-mounted display (HMD), pantalla retina virtual (VRD) y la pantalla proyectiva montada en la cabeza (HMPD).
- Hand-held: esta categoría incluye pantallas de mano y proyectores portátiles, actualmente se está trabajando en introducir estos objetivos en la AR ya que son menos intrusivos que los anteriores.
- Spatial: están colocados estáticamente en el medio, e incluyen pantalla de vídeo, pantallas ópticas espaciales, y proyectores.

Son muy diversas las areas de aplicación de la realidad aumentada, algunas de ellas son:

- Sistemas de información personal: AR puede servir como una interfaz más avanzada, natural y portable en la vida diaria.
    - Asistencia personal: una aplicación de la AR es un asistente personal que te va recordando los nombres de las personas o las tareas que tienes que hacer.
    - Navegación: se pueden proyectar imágenes de la ruta que hay que seguir, de los límites de velocidad de la vía,…
    - Touring: aportar información geográfica o histórica de un sitio que estés visitando, o superponer imágenes antiguas al ambiente real.
- Aplicaciones militares e industriales: diseño, montaje y mantenimiento son áreas donde la AR puede resultar provechosa.
- Aplicaciones médicas: enfermeras y doctores pueden beneficiarse de llevar unas gafas que les aporten información adicional de un paciente o de una patología mientras realizan su trabajo.
- Entretenimiento: creando juegos de AR o mejorando la visibilidad en eventos deportivos.
- Oficina: gestión pública, planificación urbana.
- Educación: mejora de los métodos de e-learning.

3. Kinect

Kinect es un sistema de reconocimiento del movimiento creado por Alex Kipman y desarrollado por Microsoft inicialmente para la videoconsola Xbox 360, siendo hoy en día compatible con PCs que ejecuten Windows 7 o Windows 8. El lanzamiento de Kinect se produjo el 4 de noviembre de 2010 en Norteamérica y el 10 de noviembre de 2010 en Europa.

Kinect permite a los usuarios interactuar con la consola sin necesidad de mantener contacto físico con ésta, ya que es capaz de reconocer los gestos, comandos de voz, movimientos, objetos e imágenes del entorno y del propio usuario utilizando para ello una combinación de cámaras y micrófonos que transforman los movimientos, imágenes y sonidos en acciones dentro de los videojuegos u otras aplicaciones.

El primer sensor Kinect presentado consistía en una barra horizontal de 28 cm aproximadamente conectado a una pequeña base circular, la cual disponía de un pequeño eje de articulación para ajustar la orientación del dispositivo, diseñado para ser colocado longitudinalmente por encima o por debajo de la pantalla del televisor. El dispositivo contaba con una cámara RGB (compatible con el reconocimiento facial), un sensor de profundidad (proyector de infrarrojos combinado con un sensor CMOS monocromo que permite al dispositivo ver la habitación en 3D en cualquier condición de luz), un micrófono de múltiples matrices (capaz de detectar las voces por el sonido y de extraer el ruido ambiental) y un procesador personalizado que ejecuta el software que permite a los dispositivos anteriormente mencionados realizar sus funciones y combinar la información consiguiendo así el reconocimiento vocal, facial y de movimiento.

En febrero de 2012 Microsoft lanza el kit de desarrollo software oficial para Windows, permitiendo a los desarrolladores el uso y creación de aplicaciones en Windows Usando el dispositivo Kinect.

La principal ventaja que nos aporta este SDK en la realidad aumentada es el seguimiento del esqueleto, ya que nos permite realizar body-traking de 20 puntos específicos de uno o dos usuarios.

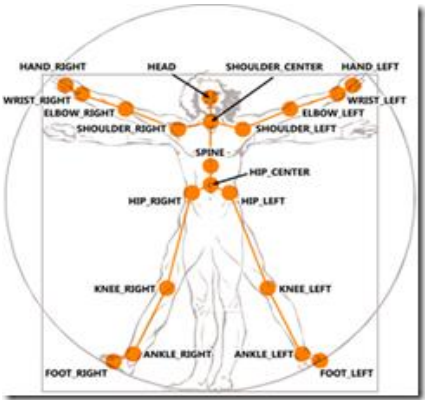

Figura 3. Puntos específicos del cuerpo humano que recoge la Kinect

III. TRABAJO PRÁCTICO

La segunda mitad de mis prácticas ha consistido en realizar una aplicación para la kinect, utilizando el SDK 1.8, para ello fue necesario descargar de la página web http://www.microsoft.com/en-us/kinectforwindowsdev/Downloads.aspx y descargar los siguientes archivos:

- Kinect for Windows SDK
- Kinect for Windows Developer Kit
- Kinect for Windows Languaje Pack (Español)
- Microsoft Visual C# 2010 (ya que programe la aplicacion en C#)

Una vez instalado todo se probó el funcionamiento de la Kinect, conectándola mediante el cable USB al ordenador (Windows 7) y ejecutando alguna aplicación de prueba del Developer Kit.

Para empezar a programar una primera aplicación es necesario crear un nuevo proyecto en Microsoft Visual, del tipo "Aplicación WPF", se nos abre una pestaña del tipo MainWindow.xaml, aquí se nos muestra como será la pantalla en la que se mostrara la ejecución de la aplicación, en mi caso la he configurado de esta forma:

```
<Window x:Class="Primera_fase.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="960" Width="660">
    <Grid>
        <Image x:Name="imageKinect" Height="480" Width="640" Margin="0,0,0,441" />
        <Image x:Name="depthKinect" Height="480" Width="640" Margin="0,441,0,0" />
        <Image Name="esqueletoKinect" Margin="68,27,72,2.8"/>
    </Grid>
</Window>
```

Se indica que la pantalla principal tendrá unas dimensiones de 960x660 y que estará formado por tres marcos, de las dimensiones indicadas y colocados en esos sitios concretos.

Ahora abrimos la pantalla MainWindow.xaml.cs y lo primero que hay que hacer es añadir la referencia a la Kinect, para ello pulsamos con el botón derecho del ratón sobre *References* y seleccionamos *Agregar referencia*, y buscamos *Microsoft.Kinect*, entonces ya podemos escribir el nuevo usinng (*using Microsoft.Kinect*) sin que la herramiento nos informe de ningún error.

Escribimos el constructor:

```
public MainWindow()
 {
     InitializeComponent();
     // Agregamos el evento de carga de la ventana
     Loaded += MainWindowLoaded;
 }
```

Y codificamos el evento de carga de la ventana:

```
void MainWindowLoaded(object sender, RoutedEventArgs e)
 {
     //Creamos un Drawing Group que sera usado para dibujar
     this.drawingGroup = new DrawingGroup();
     //Create an image Source que mostrará el esqueleto
     this.imageSource = new DrawingImage(this.drawingGroup);
     //Lo asignamos a nuestro marco definido anteriormente
     esqueletoKinect.Source = imageSource;
```

```
            // Comprobamos que tenemos un sensor conectado
            if (KinectSensor.KinectSensors.Count > 0)
            {
                //Evento ejecutado al cerrar
                Closing += MainClosing;
                // Escogemos el primer sensor kinect que tengamos conectado. Puede haber
más de un kinect conectado (miKinect ha sido creado previamente como KinectSensor
miKinect
                miKinect = KinectSensor.KinectSensors[0];
                // Habilitamos la cámara elegiendo el formato de imagen.
                miKinect.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
                miKinect.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);
                //Queremos usar el modo por defecto Mode(Human Skeleton Standing) ||
Seated(Human Skeleton Sitting Down)
                miKinect.SkeletonStream.TrackingMode = SkeletonTrackingMode.Default;
                //Nos suscribimos al método KinectSkeletonFrameReady
                miKinect.SkeletonFrameReady += KinectSkeletonFrameReady;
                // Habilitamos SkeletonStream
                miKinect.SkeletonStream.Enable();
                // Arrancamos Kinect.
                miKinect.Start();
                // Nos suscribimos al método
                miKinect.AllFramesReady += KinectAllFramesReady;
            }
        }
```

El método KinectAllFramesReady es utilizado para asignar la imagen que queramos a cada marco definido en el MainWindow.xaml, para la imagen de la cámara RGB:

```
        using (var colorFrame = e.OpenColorImageFrame())
        {
               // Si este es null no continuamos
               if (colorFrame == null) return;
               // Creamos un array de bytes del tamaño de los pixel del frame.
               byte[] pixels = new byte[colorFrame.PixelDataLength];
               //Copiamos los pixel del frame a nuestro array de bytes.
               colorFrame.CopyPixelDataTo(pixels);
               // Colocamos los pixel del frame en la imagen del xml
               int stride = colorFrame.Width * 4;
               imageKinect.Source = BitmapSource.Create(colorFrame.Width,
colorFrame.Height, 96, 96, PixelFormats.Bgr32, null, pixels, stride);
               //imageKinect es el objeto imagen colocado en el xaml
        }
```

El fragmento de código para mostrar la imagen de la cámara de profundidad es algo más complejo, porque una vez extraída la información del sensor es necesaria pintarla, para ello lo realice de dos maneras distintas, la primera mostrando la profundidad en una escala de grises, y la segunda pintando los cuerpos humanos detectados en diferentes colores en función de la distancia a la cámara a la que se encontrasen.

El método KinectSkeletonFrameReady se encarga de recoger la información de la Kinect y después pintar los puntos del cuerpo, para ello llama a otras funciones las cuales se encargan de realizar labores específicas para conseguir una mayor abstracción de cada método en el programa.

```
        void KinectSkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
         {
             //Declaro un array de Skeletons
             Skeleton[] esqueletos = new Skeleton[1];
             //Obtenemos el frame del esqueleto
             using (SkeletonFrame skeletonframe = e.OpenSkeletonFrame())
             {
                 //comprobamos que no tiene un valor nulo (ha sido abierto correctamente)
                 if (skeletonframe != null)
                 {
                     esqueletos = new Skeleton[skeletonframe.SkeletonArrayLength];
                     //Copias el array de esqueletos que contiene una colección de puntos
(joints)
                     skeletonframe.CopySkeletonDataTo(esqueletos);
                     //Si el jugador esta de pie
                     if (miKinect.SkeletonStream.TrackingMode ==
SkeletonTrackingMode.Default)
                     {
                         //Dibujas el esqueleto de pie
                         DibujarDePie(esqueletos);
                     }
                     //Si el jugador está sentado
                     else if (miKinect.SkeletonStream.TrackingMode ==
SkeletonTrackingMode.Seated)
                     {
                         //Dibujas el esqueleto sentado (10 puntos)
                         DibujarSentado(esqueletos);
                     }
                 }
             }
         }
```

La Kinect es capaz de reconocer a un usuario tanto si está de pie como si está sentado, por simplicidad indicaré aquí solo las funciones que se realizan cuando el usuario se encuentra en el primer estado.

```
        private void DibujarDePie(Skeleton[] skeletons)
         {

             using (DrawingContext dc = this.drawingGroup.Open())
             {
                 //Dibujas el fondo en forma de rectángulo, en este caso de color negro
con las dimensiones que queremos
                 dc.DrawRectangle(Brushes.Black, null, new Rect(0.0, 0.0, RenderWidth,
RenderHeight));
                 //Si el array del esqueleto no está vacio
                 if (skeletons.Length != 0)
                 {
                     //Recorrer los puntos del esqueleto
                     foreach (Skeleton skel in skeletons)
                     {
                         if (skel.TrackingState == SkeletonTrackingState.Tracked)
                         {
                             this.DrawBonesAndJoints(skel, dc);
```

```
                        }
                        else if (skel.TrackingState ==
SkeletonTrackingState.PositionOnly)
                        {
                            dc.DrawEllipse(this.centerPointBrush,
                                           null,
                                           this.SkeletonPointToScreen(skel.Position),
BodyCenterThickness, BodyCenterThickness);
                        }
                    }
                }
                //Prevenir dibujar fuera de los bordes
                this.drawingGroup.ClipGeometry = new RectangleGeometry(new Rect(0.0, 0.0,
RenderWidth, RenderHeight));
            }
        }
```

Como se observa en el código se llama a la función DrawBonesAndJoint, que se encarga una vez obtenidos los puntos característicos del cuerpo humano relacionarlos para poder así pintar las líneas que los unen:

```
        private void DrawBonesAndJoints(Skeleton skeleton, DrawingContext drawingContext)
        {
            this.DrawBone(skeleton, drawingContext, JointType.Head,
JointType.ShoulderCenter);
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter,
JointType.ShoulderLeft);
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter,
JointType.ShoulderRight);
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter,
JointType.Spine);
            this.DrawBone(skeleton, drawingContext, JointType.Spine,
JointType.HipCenter);
            this.DrawBone(skeleton, drawingContext, JointType.HipCenter,
JointType.HipLeft);
            this.DrawBone(skeleton, drawingContext, JointType.HipCenter,
JointType.HipRight);
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderLeft,
JointType.ElbowLeft);
            this.DrawBone(skeleton, drawingContext, JointType.ElbowLeft,
JointType.WristLeft);
            this.DrawBone(skeleton, drawingContext, JointType.WristLeft,
JointType.HandLeft);
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderRight,
JointType.ElbowRight);
            this.DrawBone(skeleton, drawingContext, JointType.ElbowRight,
JointType.WristRight);
            this.DrawBone(skeleton, drawingContext, JointType.WristRight,
JointType.HandRight);
            this.DrawBone(skeleton, drawingContext, JointType.HipLeft,
JointType.KneeLeft);
            this.DrawBone(skeleton, drawingContext, JointType.KneeLeft,
JointType.AnkleLeft);
            this.DrawBone(skeleton, drawingContext, JointType.AnkleLeft,
JointType.FootLeft);
            this.DrawBone(skeleton, drawingContext, JointType.HipRight,
JointType.KneeRight);
```

```
            this.DrawBone(skeleton, drawingContext, JointType.KneeRight,
JointType.AnkleRight);
            this.DrawBone(skeleton, drawingContext, JointType.AnkleRight,
JointType.FootRight);

            foreach (Joint joint in skeleton.Joints)
            {
                Brush drawBrush = null;

                if (joint.TrackingState == JointTrackingState.Tracked)
                {
                    drawBrush = this.trackedJointBrush;
                }
                else if (joint.TrackingState == JointTrackingState.Inferred)
                {
                    drawBrush = this.inferredJointBrush;
                }

                if (drawBrush != null)
                {
                    drawingContext.DrawEllipse(drawBrush, null,
this.SkeletonPointToScreen(joint.Position), JointThickness, JointThickness);
                }
            }
        }
```

Para dibujar este "hueso" que une cada articulación se utiliza la siguiente función:

```
       private void DrawBone(Skeleton skeleton, DrawingContext drawingContext, JointType
jointType0, JointType jointType1)
        {
            Joint joint0 = skeleton.Joints[jointType0];
            Joint joint1 = skeleton.Joints[jointType1];

            if (joint0.TrackingState == JointTrackingState.NotTracked ||
joint1.TrackingState == JointTrackingState.NotTracked)
            {
                return;
            }
            if (joint0.TrackingState == JointTrackingState.Inferred &&
joint1.TrackingState == JointTrackingState.Inferred)
            {
                return;
            }
            Pen drawPen = this.inferredBonePen;
            if (joint0.TrackingState == JointTrackingState.Tracked &&
joint1.TrackingState == JointTrackingState.Tracked)
            {
                drawPen = this.trackedBonePen;
            }
            drawingContext.DrawLine(drawPen, this.SkeletonPointToScreen(joint0.Position),
this.SkeletonPointToScreen(joint1.Position));
        }
```

Y por ultimo para poder pintar el esqueleto, debemos saber exactamente las coordenadas del punto

```
        private Point SkeletonPointToScreen(SkeletonPoint skelpoint)
        {
            DepthImagePoint depthPoint =
this.miKinect.CoordinateMapper.MapSkeletonPointToDepthPoint(skelpoint,
DepthImageFormat.Resolution640x480Fps30);
            return new Point(depthPoint.X, depthPoint.Y);
        }
```

Una vez queramos detener nuestra aplicación, debemos tener un método que se encargue de detener todo de una forma segura:

```
        void MainClosing(object sender, EventArgs e)
        {
            // Al cerrar paramos Kinect
            if (miKinect != null)
            {
                miKinect.Stop();
            }
        }
```

Ya que está todo codificado se prueba la aplicación, dando como resultado una pantalla con tres marcos, uno arriba que muestra la imagen obtenida de la cámara RGB, uno en la parte inferior que muestra los datos de la cámara de profundidad (bien en escala de grises o coloreando los cuerpos de las personas) y un último marco en el centro superpuesto a los otros dos en el que se observan los puntos del esqueleto de la persona a la que la cámara esta captando dibujados en color verde y unidos entre si formando un "esqueleto".

IV. MEJORAS FUTURAS

La complejidad de este trabajo practico ha sido la de empezar desde cero un proyecto nuevo, una vez que se tiene esta base se puede avanzar de una manera mucho más rápida en la realización de aplicaciones más complejas, una posible mejora sería el intentar superponer al esqueleto dibujado únicamente con líneas una imagen de un avatar y que este avatar pudiese interactuar con objetos que le apareciesen como una pelota.

V. CONCLUSIONES

La línea de trabajo seguida ha sido la correcta, primero fue necesario un entendimiento de que es la realidad aumentada, como está presente en nuestras vidas y como podría llegar a mejorarlas, estudiando los medios en los que se puede conseguir, con que dispositivos y en que ámbitos, pero teniendo presente que la interacción hombre-máquina debe ser lo más natural posible es necesario encontrar métodos que no sean intrusivos, por lo que no hay nada menos intrusivo que no llevar nada, simplemente que unos sensores recojan información y traten esa información para hacer la vida más fácil; estos sensores nos los aporta la Kinect.

VI. REFERENCIAS

1. Karam M, "A framework for research and design of gesture-based human computer interactions", PhD Thesis, University of Southampton, 2006
2. D. W. F. van Krevelen, R. Poelman, "A Survey of Augmented Reality Technologies, Applications and Limitations", Systems Engineering Section, Delft University of Technology, Delft, The Netherlands, 26 enero 2010
3. J. Carmigniani, "Augmented reality technologies, systems and applications", Florida Atlantic University, 14 diciembre 2010
4. Hung-Lin Chi, "Research trends and opportunities of augmented reality applications in architecture, engineering, and construction" , National Taiwan University, agosto 2013
5. Derek McColl, Goldie Nejat, Z. Zhang, "Affect Detection from Body Language during Social HRI", Univ. of Toronto, Canada, 2012
6. Derek McColl, Goldie Nejat, Z. Zhang, "Human Body Pose Interpretation and Classification for Social Human-Robot Interaction", Univ. of Toronto, Canada, 2011
7. Derek McColl, Goldie Nejat, Z. Zhang, "Meal-Time with a Socially Assistive Robot and Older Adults at a Long-term Care Facility", Univ. of Toronto, Canada, 2013
8. Ginevra Castellano, Santiago D. Villalba, Antonio Camurri,, "Recognising Human Emotions from Body Movement and Gesture Dynamics" University College Dublin, Ireland, 2007
9. Andrea Kleinsmith, Nadia Bianchi-Berthouze, "Affective Body Expression Perception and Recognition: A Survey", 2013
10. Ales Prochazka, Miroslav Kubicek, Ales Pavelka, "Multicamera Systems in the Moving Body Recognition", Prague Institute of Chemical Technology, Czech Republic, 2006
11. Fei Xie, Guili Xu, Yuehua Cheng, Yupeng Tian "An Improved Thinning Algorithm for Human Body Recognition" Nanjing University of Aeronautics & Astronautics, China, 2009
12. Fang Huang, Fa-Niu Xu and Xiao-Ping Fan "Relational Match with Parallel Interpretation Tree for Recognition of Part-Based Human Body", Central South University, China, 2010
13. E. Davaasambuu, Chia-Chi Chiang, J. Y.Chiang, Yung-Fu Chen, S. Bilgee, "A Microsoft Kinect based virtual rehabilitation system" Taiwan University of Science and Technology, 2012
14. Jiawen Chen, Shahram Izadi, Andrew Fitzgibbon, "KinÊ tre: Animating the World with the Human Body", Microsoft Research Cambridge, UK, 2012
15. J. L. Raheja, R. Shyam, Umesh. Kumar, P. B. Prasad, "Real-Time Robotic Hand Control using Hand Gestures", Digital Systems Group Central Electronics Engineering Research Institute (CEERI) Rajasthan, India, 2010
16. Hong-xiang DUAN, Qiu-yu ZHANG, Wei MA, "An Approach to Dynamic Hand Gesture Modeling and Real-time Extraction", Lanzhou University, China, 2011
17. Siddharth S. Rautaray, Anupam Agrawal "Design of Gesture Recognition System for Dynamic User Interface" 2011
18. Siddharth S. Rautaray, Anupam Agrawal "Vision based hand gesture recognition for human computer interaction: a survey" 2012
19. Richard Nolberto Rojas Bello "Richard Nolberto Rojas Bello," UAM, 2009
20. C. Tanchotsrinon, S.Phimoltares and S. Maneeroj "Facial expression recognition using graph-based features and artificial neural networks" Chulalongkorn University, Bangkok, 10330, Thailand, 2011
21. H.K Ekenel, J. Stallkamp, H. Gao, M. Fischer, R. Stiefelhagen "Face recognition for smart interactions", Universitat Karlsruhe (TH) 76131, Karlsruhe, Germany, 2007
22. Ming Zhao, Tat-Seng Chua "Markovian Mixture Face Recognition with Discriminative Face Alignment", 2008
23. Josh Harguess and J. K. Aggarwal, "A Case for the Average-Half-Face in 2D and 3D for Face Recognition", The University of Texas at Austin, 2009
24. Manzoor Ahmad Lone, S. M. Zakariya and Rashid Ali "Automatic Face Recognition System by Combining Four Individual Algorithms", Baba Ghulam Shah Badshah University y Aligarh Muslim University, India 2011
25. G. Prabhu Teja, S. Ravi, "Face Recognition using Subspaces Techniques", Pondicherry University Pondicherry, India, 2012
26. Shahram Izadi, Microsoft Et al, "KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera", 2011

VII. ANEXO I

Tabla clasificatoria de artículos en función de las tecnologías aplicadas, sistemas de detección, autores, universidades y años.

Título	Descripción del sistema	Tecnologías aplicadas	Sistema de detección	Autores/Universidad	Año
Affect Detection from Body Language during Social HRI	Sistema de clasificación automática del estado afectivo de una persona desde su postura corporal estática	HRI (Brian 2.0), Kinect	Lenguaje corporal, posturas.	Derek McColl, Goldie Nejat, Z. Zhang, Univ. of Toronto, Canada http://asblab.mie.utoronto.ca/research	2012
Human Body Pose Interpretation and Classification for Social Human-Robot Interaction					2011
Meal-Time with a Socially Assistive Robot and Older Adults at a Long-term Care Facility					2013
Recognising Human Emotions from Body Movement and Gesture Dynamics	Sistema de reconocimiento de estados emocionales en función de la pose y el movimiento corporal	EyesWeb platform, EyesWeb Expressive Gesture Processing Library	Lenguaje corporal.	Ginevra Castellano, Santiago D. Villalba, Antonio Camurri, University College Dublin, Ireland	2007
Affective Body Expression Perception and Recognition: A	Survey acerca del reconocimiento de la expresión corporal		Expresion corporal	Andrea Kleinsmith, Nadia Bianchi-Berthouze	2013

Survey					
Multicamera Systems in the Moving Body Recognition	Sistema de reconocimiento del cuerpo en movimiento y su modelización matemática	Dos cámaras DragonFly y PC con MATLAB Image Acquisition Toolbox	Movimiento corporal	Ales Prochazka, Miroslav Kubicek, Ales Pavelka, Prague Institute of Chemical Technology, Czech Republic	2006
An Improved Thinning Algorithm for Human Body Recognition	Métodos de reconocimiento del cuerpo para los sistemas de vigilancia visual	Uso de cuatro algoritmos: OPTA, Zhang, Rosenfeld y una propuesto por los autores	Poses corporales y específicamente de las piernas humanas	Fei Xie, Guili Xu, Yuehua Cheng, Yupeng Tian, Nanjing University of Aeronautics & Astronautics, China	2009
Relational Match with Parallel Interpretation Tree for Recognition of Part-Based Human Body	Reconocimiento de partes del cuerpo humano recorriendo las ramas de un árbol.		Cuerpo humano por partes	Fang Huang, Fa-Niu Xu and Xiao-Ping Fan, Central South University, China	2010
A Microsoft Kinect based virtual rehabilitation system	Rehabilitación física	Kinect	Posición, gestos	E. Davaasambuu, Chia-Chi Chiang, J. Y.Chiang, Yung-Fu Chen, S. Bilgee, Taiwan University of Science and Technology, http://image.cse.nsysu.edu.tw	2012
KinÊ tre: Animating the World with the Human Body	Sistema que permite escanear un objeto y que éste interactúe con el usuario	Kinect, sistema KinectFusion	Movimiento corporal	Jiawen Chen, Shahram Izadi, Andrew Fitzgibbon, Microsoft Research Cambridge, UK	2012
Real-Time Robotic Hand Control using Hand Gestures	El control de una mano robótica o un robot individual simplemente mostrando gestos de la mano delante de una cámara	HRI	Gestos con las manos	J. L. Raheja, R. Shyam, Umesh. Kumar, P. B. Prasad, Digital Systems Group Central Electronics Engineering Research Institute (CEERI) Rajasthan,	2010

				India http://www.ceeri.ernet.in/	
An Approach to Dynamic Hand Gesture Modeling and Real-time Extraction	Nuevo modelo dinámico reconocimiento de gestos de la mano y su extracción en tiempo real mediante la combinación de tres factores (postura inicial de la mano, trayectoria intermedia y postura final)	Algoritmo de detección de contorno, segmentación del color de la piel, algoritmo "Mean-shift" y de traking	Gestos con las manos	Hong-xiang DUAN, Qiu-yu ZHANG, Wei MA, Lanzhou University, China http://www.lzu.edu.cn/notice/English/default.htm	2011
Design of Gesture Recognition System for Dynamic User Interface	Sistema simple y natural para l interacción gestual entre el usuario y el ordenador ofreciendo una interfaz de usuario dinámica. El sistema de reconocimiento gestual utiliza técnicas de procesamiento, detección, segmentación y traking	Librería OpenCV, ordenador con procesador de 1.99 GHz, web cam con resolución de 320x240	Gestos con las manos estáticos y dinámicos	Siddharth S. Rautaray, Anupam Agrawal	2011
Vision based hand gesture recognition for human computer interaction: a survey	Análisis comparativo del área		Gestos con las manos		2012
Richard Nolberto Rojas Bello,	Descripción de los distintos métodos para el reconocimiento de expresiones faciales		Expresiones faciales	Richard Nolberto Rojas Bello, UAM	2009
Facial expression recognition using graph-based features and artificial neural networks	Sistema que localiza los puntos de la región de cara para formar grafos y entrenar las redes neuronales a reconocer las emociones		Emociones faciales	C. Tanchotsrinon, S.Phimoltares and S. Maneeroj, Chulalongkorn University, Bangkok, 10330, Thailand	2011
Face recognition for smart interactions	Se presentan tres sistemas de reconocimiento facial basados en un algoritmo que utiliza los coeficientes de la transformada discreta del coseno, estos	Webcam y ordenador	Facial	H.K Ekenel, J. Stallkamp, H. Gao, M. Fischer, R. Stiefelhagen, Universitat Karlsruhe (TH) 76131, Karlsruhe, Germany,	2007

	sistemas son: “door monitoring system" (observa la entrada a una habitación e identifica a los sujetos cuando éstos están entrando en esa habitación),"portable face recognition system" (usado en cualquier tipo de escenario) y "3D face recognition system" (reconocimiento automático de los datos en 3D			http://isl.anthropomatik.kit.edu/english/	
Markovian Mixture Face Recognition with Discriminative Face Alignment	Reconocimiento facial basandose en procesos estocásticos de Markov		Facial	Ming Zhao, Tat-Seng Chua	2008
A Case for the Average-Half-Face in 2D and 3D for Face Recognition	Comparativa de seis algoritmos en la utilizacion de reconocimiento facial “Average-half”	Eingfaces, MPCA, MPCA with Linear Discriminant Analysis (MPCALDA), Fisherfaces or Linear Discriminant Analysis (LDA), Independent Component Analysis (ICA), Support Vector Machine (SVM)	Facial	Josh Harguess and J. K. Aggarwal, The University of Texas at Austin	2009
Automatic Face Recognition System by Combining Four Individual Algorithms	Sistema de reconocimiento facial basado en una combinación de cuatro técnicas	Técnica Principal Component Analysis (PCA), Discrete Cosine Transform	Facial	Manzoor Ahmad Lone, S. M. Zakariya and Rashid Ali, Baba Ghulam Shah Badshah University y Aligarh Muslim University,	2011

		(DCT), Template Matching using Correlation (Corr) and Partitioned Iterative Function System (PIFS)		India	
Face Recognition using Subspaces Techniques	Visión de las diferentes técnicas de reconocimiento facial centrándose en las técnicas de subespacio, investigando el preprocesado de la imagen con el fin de reducir las tasas de error.	Métodos de reconocimiento: enfoque basado en plantilla geométrica, enfoque por etapas, enfoque basado en modelos y enfoque en redes neuronales.	Facial	G. Prabhu Teja, S. Ravi, Pondicherry University Pondicherry, India	2012
KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera	Novel GPU pipeline tque logra seguimiento 3D, reconstrucción, segmentación, representación e interacción, todo en tiempo real, utilizando sólo una cámara de y hardware de gráficos. Escaneo de objetos de bajo coste y AR avanzado. Nuevos métodos de segmentaci, seguimiento y reconstrucción dinámica de usuarios con la escena.	Kincet, depth map, AR-ML		Shahram Izadi, Microsoft Et al.: UK, Canada	2011