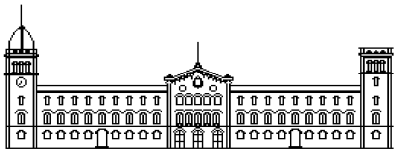


Trabajo final de carrera

INGENIERIA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Facultad de Matemáticas
Universidad de Barcelona

RGBD Análisis para el Reconocimiento Automático de Acciones con Kinect.

Gabriel Domínguez Cisternes

Director: Sergio Escalera y Miguel Reyes.
Realizado en: Dept. Matemática Aplicada i Análisis.
Barcelona, 30 de junio de 2011

Agradecimientos.

Este proyecto no hubiera sido posible sin la ayuda y el apoyo de muchas personas. Me gustaría por tanto dedicárselo a todas ellas:

A Sergio Escalera y Miguel Reyes, que siempre han encontrado el momento para discutir y solucionar las dudas que han ido surgiendo a través de las etapas más difíciles del proyecto.

A todos mis familiares, amigos y compañeros que se han prestado a posar ”voluntariamente” numerosas veces delante de la cámara a repetir gestos una y otra vez.

A mis compañeros Daniel Ferreiro y Juan Carlos Ortega por la paciencia y la ayuda prestada a retoques y detalles de código, a Mireia Diaz por las horas invertidas en cuestiones de redacción y revisión de textos.

Y al resto de amigos, compañeros, y profesores que aunque no hayan formado parte de este proyecto, han formado parte de esta gran experiencia a lo largo de estos 3 últimos años.

Abstract.

This project implements and analyzes the effectiveness Dynamic Time Warping algorithm (DTW), applied to the classification of gestures. To do this, the gestures are defined by a set of positions originally composed by a group of 15 points in space (3D). To obtain these points using the Microsoft Kinect device, device with the ability to record image, get the distance from the device to surrounding objects and other features. To implement the algorithm using the programming language C + + and makes use of the Open library (which handles communication between the application and the device) and OpenCV (used in the process of normalization).

Resum.

En aquest projecte s'implementa i analitza l'efectivitat de l'algorisme Dynamic Time Warping (DTW), aplicat a la classificació de gestos. Per això, els gestos vénen definits per un conjunt de postures compostes originalment per un grup de 15 punts en l'espai (3D). Per a l'obtenció d'aquests punts s'utilitza el dispositiu Microsoft Kinect, dispositiu amb la capacitat de gravar imatge, obtenir la distància des del dispositiu fins als objectes de l'entorn i altres funcions. Per a la implementació de l'algorisme s'utilitza el llenguatge de programació C + + i es fa ús de les llibreries Open (que s'encarrega de la comunicació entre l'aplicació i el dispositiu) i OpenCV (que s'utilitza en els processos de normalització).

Resumen.

En este proyecto se implementa y analiza la efectividad del algoritmo Dynamic Time Warping (DTW), aplicado a la clasificación de gestos. Para ello, los gestos vienen definidos por un conjunto de posturas compuestas originalmente por un grupo de 15 puntos en el espacio (3D). Para la obtención de estos puntos se utiliza el dispositivo Microsoft Kinect, dispositivo con la capacidad de grabar imagen, obtener la distancia desde el dispositivo hasta los objetos del entorno y otras funciones. Para la implementación del algoritmo se utiliza el lenguaje de programación C++ y se hace uso de las librerías Open (que se encarga de la comunicación entre la aplicación y el dispositivo) y OpenCV (que se utiliza en los procesos de normalización).

Índice general

CAPÍTULO 1

Introducción

El objetivo de este proyecto es implementar y analizar un algoritmo que nos permita el reconocimiento automático de acciones realizadas por un usuario. Para ello, utilizamos un dispositivo capaz de visualizar el entorno captando información sobre la profundidad, en un sistema conocido como RGBD (Red-Green-Blue-Depth).

1.1. Contexto

Para la realización del proyecto, es necesario aclarar dos conceptos que son:

- Qué es una acción y que tipo de acciones se pueden reconocer.
- Qué información es necesaria y como se obtiene.

1.1.1. Acciones

Se toma como acción cualquier movimiento realizado por una persona, este movimiento debe ser identificable. Es decir que se pueda entender con claridad a simple vista, como pueden ser:

- Saludar con la mano derecha.
- Señalar al frente con la mano derecha.
- Aplaudir.

- Saltar.
- Agacharse.

Como se puede observar, son acciones simples, claras y que a priori todo el mundo realiza de una forma similar. En ningún momento se pueden utilizar acciones que tengan algún comportamiento demasiado aleatorio (e.g abrochar una camisa, no se puede predecir si se empezará por el botón de arriba, por el de abajo, un botón de arriba luego el de abajo, etc.), Tampoco se pueden utilizar acciones que requieran interactuar con el entorno o con otros usuarios, ya que los algoritmos que se plantean en este proyecto analizan los sujetos de forma individual.

1.1.2. Dispositivo

Como captador de profundidad, se utiliza el sensor de Microsoft Kinect (Figura 1.1). Este dispositivo fue anunciado por primera vez el 1 de junio de 2009 en la Electronic Entertainment Expo 2009 como "Project Natal". Kinect es una barra horizontal de aproximadamente 23cm conectada a una pequeña base circular con un eje de articulación tipo rótula, y está diseñado para ser colocado longitudinalmente por encima o por debajo de una pantalla de visualización. El dispositivo es capaz de captar información de color (como una webcam convencional), sonido y la distancia que hay desde el objetivo hasta los objetos del entorno (paredes, mobiliarios, personas, etc.). Las especificaciones técnicas del dispositivo se describen en el apartado 2.1.

Figura 1.1: Dispositivo Kinect.

1.2. Motivación

Hoy en día la tendencia a automatizar tareas está en alza y las nuevas tecnologías van cobrando poder en la vida cotidiana de las personas. Generalmente, al pensar en nuevas tecnologías pensamos en dispositivos que hacen tareas muy complejas aplicando algoritmos incomprensibles para la mayoría de la gente, pero esto no siempre es así. En este caso, aprovechando una nueva tecnología como son las cámaras con sensor de profundidad, aplicamos un algoritmo conocido como "Dynamic Time Warping" (DTW) que tiene una complejidad media a una tarea simple como es el reconocimiento de un gesto, y nos centramos en ver la efectividad del algoritmo, sus puntos fuertes, sus puntos flacos y sus posibles mejoras.

1.3. Estado del Arte

En relación a nuestro objetivo, no hay demasiados avances ya que los dispositivos con sensores de profundidad son relativamente nuevos, y hasta hace poco tiempo no se disponía de un framework capaz de trabajar con los dispositivos de una forma cómoda.

1.3.1. Reconocimiento de gestos

Hasta el momento, el reconocimiento de gestos es una rama bastante amplia de la visión por computador, trata el reconocimiento de gestos como un conjunto de técnicas de procesado de imágenes aplicadas a una serie de datos obtenidos a través de una cámara convencional en un formato de representación clásico (RGB, YUV, B/W, etc.) y haciendo uso generalmente de la librería OpenCV[1][2]. Posterior al procesado de imagen se hace uso de tecnicas de reconocimiento de patrones para el analisis de gestos, tales como Hidden Markov Models[3][4] o Dynamic Time Warping[5][6].

Las principales aplicaciones del reconocimiento de gestos, están orientadas al control de interfaces hombre maquina, aplicaciones en robótica, realidad virtual, al procesado de gestos y traducción a voz del lenguaje de signos, etc.

Como ejemplos de aplicación se pueden observar las siguientes:

- Control del Pc con webcam - La mano como mouse[7]: Esta aplicación permite substituir el control clásico del mouse, adaptándolo a los movimientos de la mano del usuario.
- Reconocimiento de gestos - Números con las manos[8]: Esta aplicación incluye el uso de la cámara Kinect y sirve para reconocer un número representado con los dedos de una mano.

1.3.2. Aplicaciones con sensores de profundidad

Existen aplicaciones con sensores de profundidad orientadas a fines muy diversos. Como principal uso de la cámara Kinect se puede ver cualquier aplicación (juego) de la empresa Xbox 360, donde se puede observar fácilmente el potencial del dispositivo en el momento de detectar y realizar el seguimiento de una persona.

También podemos ver aplicaciones más modestas hechas por usuarios que no pertenecen a grandes empresas, como por ejemplo:

- AppSinbad[9]: Es un una aplicación que detecta a una persona y la reconoce (calibra su posición) para posteriormente seguir su movimiento y trasladarlo a un avatar virtual, que reproduce los movimientos de la persona.
- Aquila iCub teleoperation[10]: De una forma similar a la de la aplicación anterior, esta aplicación traslada el movimiento de la persona a un robot.
- 3D drawing with the Kinect[11]: Esta aplicación haciendo un uso combinado de la librería OpenCV y la cámara Kinect, permite la creación de dibujos en 3D.

1.4. Propuesta

En este proyecto se fijan dos objetivos directos, el primero de ellos se basa en el análisis de efectividad del algoritmo DTW sobre un conjunto de cinco acciones determinadas y el segundo se trata de realizar una aplicación real sobre datos obtenidos de niños con Trastorno por déficit de atención con hiperactividad (TDAH). Todos los datos de las aplicaciones serán adquiridos mediante el dispositivo Kinect.

Para poder llegar a estos objetivos son necesarios los siguientes puntos:

1.4.1. Análisis de Kinect

Antes de empezar a trabajar en la aplicación existe un trabajo previo, consiste en averiguar las características del dispositivo Kinect. Se estudian las características técnicas tanto a nivel de hardware como a nivel de software (librerías, códigos de ejemplo, etc.).

1.4.2. Calibración automática

Para poder realizar el seguimiento de una persona mediante el dispositivo Kinect observamos que es preciso pasar por una fase de calibración; una parte del proyecto se centrará exclusivamente en automatizar este proceso, de manera que será posible realizar el seguimiento de personas en grabaciones en las que las personas participantes no realizan la fase de calibración.

1.4.3. Análisis de acciones

Para el análisis de las acciones implementamos el algoritmo DTW (descrito en el apartado 3.4). A grandes rasgos este algoritmo calcula la similitud entre dos señales (en nuestro caso una acción) y dependiendo del valor obtenido se puede decidir si una acción es suficientemente similar a otro como para asumir que es la misma.

1.4.4. Definición de una base de datos

Para poder cumplir el primer objetivo es necesario definir una base de datos que debe contener los datos capturados con la cámara Kinect, referentes al conjunto de acciones que se pretende estudiar, realizadas por un conjunto amplio de usuarios modelo. Definido en detalle en el apartado 4.1.1.

1.4.5. Aplicación TDAH

Finalmente se diseña una aplicación que pone en práctica todo lo anteriormente descrito para ver el funcionamiento sobre un caso real como es el de los niños con TDAH. Esta aplicación consta de la extracción de puntos sobre unos vídeos grabados y búsqueda de un gesto determinado sobre los datos.

1.5. Estructura de la memoria

El contenido de la memoria se divide en cinco bloques, el primero corresponde al bloque actual (Introducción) que muestra una visión general del contenido del proyecto.

Capítulo 2 Kinect: Representación RGBD.

En este bloque se muestran las características del dispositivo Kinect desde dos puntos de vista diferentes: por un lado se muestran las características a nivel de hardware (dispositivos que conforman la Kinect, especificaciones técnicas,etc.) y por otro se muestra el contenido que hace referencia al software (drivers, frameworks,etc.).

Capítulo 3 Metodología.

En este bloque se muestra la manera en la que trabaja el proyecto, se describe de forma detallada cada una de las partes del sistema. (Calibración automática, detección de la persona, descripción de las acciones y reconocimiento con DTW).

Capítulo 4 Resultados.

En este bloque se muestran los resultados obtenidos, ademas se muestra una explicación completa de cómo se obtienen.

- Datos: describimos como están definidos lo datos con los que se trabaja, tanto la base de datos usada en el primer objetivo como los datos utilizados en la aplicación a los niños con TDAH.
- Métodos: se muestra los datos de configuración del algoritmo DTW y como se obtienen.
- Validación: se describe en detalle como se ha hecho la validación de algoritmo.
- Finalmente hay una descripción de los resultados del experimento 1 y del experimento 2.

CAPÍTULO 2

Kinect: Representación RGBD

2.1. Hardware

El dispositivo Kinect cuenta con las siguientes especificaciones de fábrica:

2.1.1. Sensores

- Lentes de color (WebCam tradicional VGA 640x480).
- Sensor de profundidad (Parte imprescindible para la realización de este proyecto).
- Micrófono multi-arreglo (Permite ser usado en salas de Chat sin hacer uso de auriculares).
- Ajuste de sensor con su motor de inclinación (Permite el ajuste de la cámara de forma mecánica).

2.1.2. Campo de visión

- Campo de visión horizontal: 57 grados.
- Campo de visión vertical: 43 grados.
- Rango de inclinación física: ± 27 grados (motor de inclinación).
- Rango de profundidad del sensor: 1,2 – 3,5 metros (optimo).

2.1.3. Data Streams (Flujo de datos)

- 320 × 240 a 16 bits de profundidad @ 30fps (ajustable por software a 15fps).
- 640 × 480 32-bit de color @30fps (ajustable por software a 15fps y a 320 x 240 de resolución).
- Audio de 16-bit @ 16 Khz.

2.1.4. Sistema de Seguimiento

- Rastrea hasta 6 personas, incluyendo 2 jugadores activos.
- Rastrea 20 articulaciones por jugador activo.
- Capacidad para mapear jugadores activos en Live Avatars.

2.1.5. Esquema de dispositivos de Kinect.

Podemos ver la distribución de los dispositivos citados anteriormente en la figura2.1.

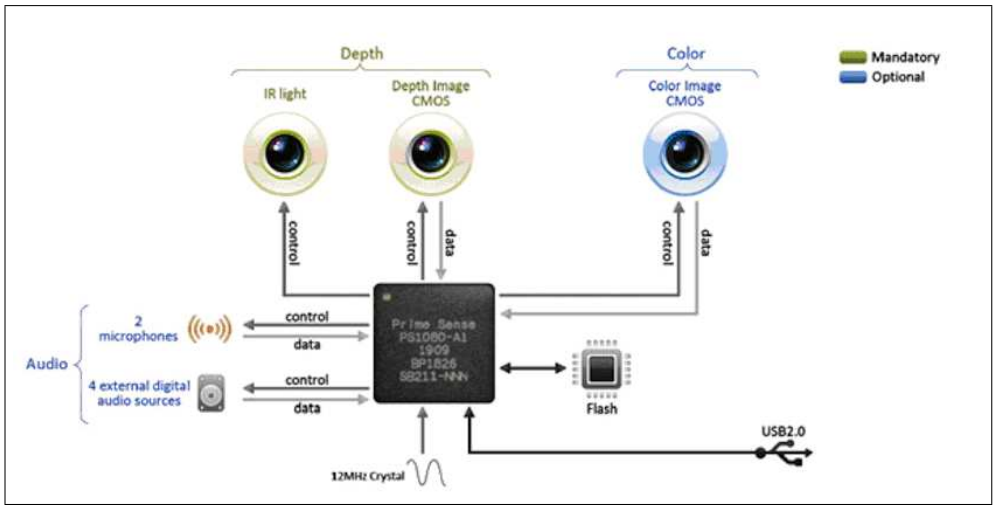


Figura 2.1: Esquema dispositvos Kinect.

2.1.6. Como capta la profundidad Kinect

El proceso de captación de la profundidad consta de cinco fases tal como muestra la figura 2.2:

1. Disponemos de un entorno.
2. Kinect mediante su dispositivo de infrarrojos lanza miles haces de luz.
3. La luz invisible rebota en el entorno.
4. El sensor CMOS capta la luz rebotada por el entorno.
5. El procesador interno de la Kinect calcula el tiempo que ha tardado en retornar la luz y nos crea un mapa de profundidad.

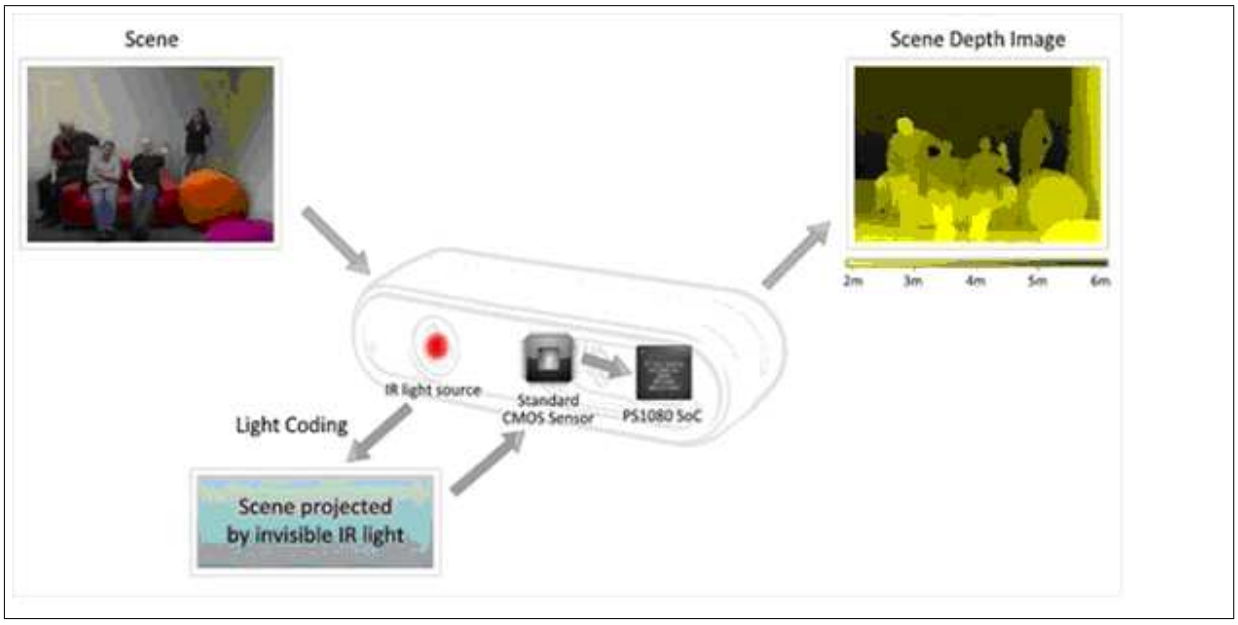


Figura 2.2: Esquema captación de profundidad del dispositivo Kinect.

2.2. Software

Para trabajar con Kinect necesitamos dos clases de software, el controlador que se encarga de la comunicación y el control del dispositivo y por otra parte el Framework que nos permite realizar la aplicación según nuestras necesidades.

2.2.1. Controlador

Para el desarrollo del proyecto se ha utilizado el controlador desarrollado por la empresa PrimeSense[12], concretamente la versión:

- SensorKinect-Win32-5.0.0

2.2.2. Framework

Actualmente existen diferentes formas de trabajar con el sensor Kinect, las más utilizadas son:

- OpenNI[13].
- OpenKinect[14].

Aunque también existe una SDK oficial de Microsoft en versión Beta hasta el momento.

- Kinect for Windows SDK beta[15].

2.2.2.1. OpenNI

Para la realización del proyecto, se decide utilizar OpenNI ya que es la que mejor se ajusta a nuestras necesidades. Inicialmente utilizamos la versión "OpenNI-Bin-Win32-v1.0.0.25" y actualizamos a la versión "OpenNI-Win32-1.1.0.38-Dev" ya que implementa nuevas funcionalidades interesantes para el desarrollo del proyecto.

Por otro lado utilizamos la API de NITE[16] versión "NITE-Bin-Win32-v1.3.0.18", que junto con OpenNI será la que nos permita hacer el seguimiento de una persona, acceder a los puntos de las articulaciones, etc.

Una vez instalado el software (ver Anexo B: Instalación OpenNI) podemos ver varias aplicaciones de muestra que son útiles para entender los conceptos básicos del funcionamiento de la librería.

2.2.2.2. Conceptos básicos OpenNI

OpenNi es una librería que permite la comunicación entre una aplicación y los sensores de la cámara Kinect, ofreciendo una variedad de funcionalidades que permiten la creación de aplicaciones a necesidad del usuario. El lenguaje de programación utilizado es C++.

Se puede observar en la figura 2.3 el modelo de capas de la librería. En la capa de aplicación se encuentra la aplicación que realiza el usuario, la capa del medio contiene las interfícies de comunicación entre los sensores del dispositivo y los componentes intermedios, que son los que analizan la información obtenida por los sensores. Finalmente se observa la capa donde se encuentran físicamente los sensores.

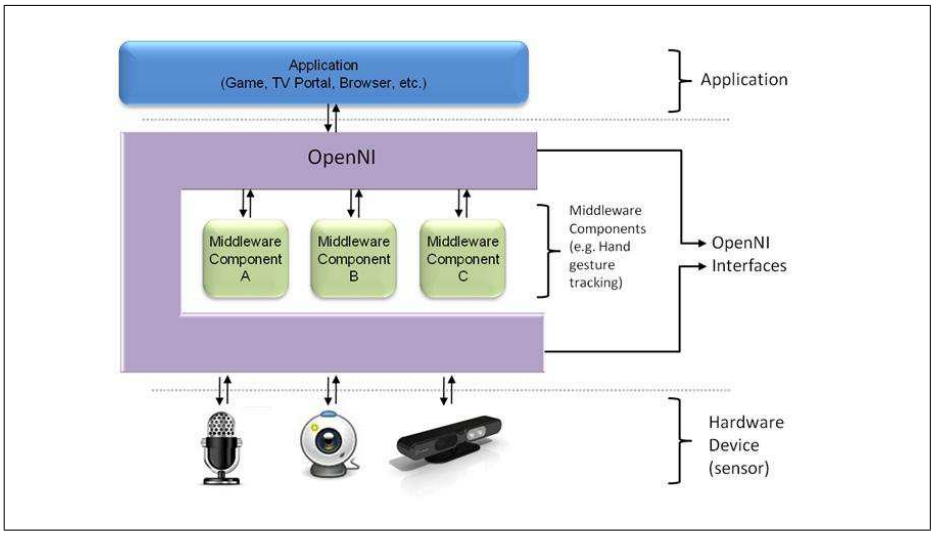


Figura 2.3: Modelo de capas de OpenNI.

OpenNI distribuye los sensores y los componentes intermedios en módulos de manera que al realizar una aplicación se utilizarán sólo aquellos que sean necesarios. Para el uso de módulos, hay que crear lo que OpenNI llama Production-Nodes, que son un conjunto de componentes que tienen el papel de productores de los datos necesarios para las aplicaciones. Cada nodo de producción encapsula la funcionalidad que se refiere a la generación del tipo de datos específico. Estos nodos de producción son los elementos fundamentales de la interfaz de OpenNI. Sin embargo, la API de los nodos de producción sólo se define en el lenguaje. La lógica de la generación de datos deben ser implementadas por los módulos que se conectan a OpenNI. En la documentación de OpenNI se describen todos los tipos que existen. A continuación se describen los que se utilizan en la apli-

cación.

Nodos de producción relativos a sensores:

- Depth Generator: Este nodo genera un mapa de profundidad del entorno, para ello necesita tener conectado un sensor de profundidad compatible con OpenNI.
- Image Generator: Este nodo captura la imagen en color del entorno, para ello necesita tener conectado un sensor de color compatible con OpenNI.

Nodos de producción relativos a componentes intermedios:

- User Generator: Genera una representación total o parcial de un cuerpo en un escenario 3D.

2.2.2.3. Sample de partida de la aplicación

Como punto de inicio en la aplicación se coge como referencia el código de ejemplo de la librería OpenNI NiUserTracker, que implementa las funcionalidades de creación e inicialización de nodos. Además implementa la detección de usuarios y la posibilidad de calibrarlos para posteriormente hacer el seguimiento del cuerpo.

2.2.2.4. Diferencia entre versiones OpenNI

El principal motivo del cambio de versión de 1.0.0.25 a la versión 1.1.0.38 es la aparición de nuevas funcionalidades referentes al proceso de calibración y seguimiento de personas:

- Posibilidad de guardar en un fichero los datos referentes a la calibración de una persona concreta.
- Posibilidad de cargar los datos guardados anteriormente sobre una persona y aplicarlos a otra persona, evitando así pasar por la fase de calibración.

CAPÍTULO 3

Metodología

3.1. Calibración automática

Al empezar a trabajar surge la necesidad de encontrar un método que permita calibrar a un usuario sin necesidad de pasar por la fase de calibración clásica, ya que en las fuentes de datos que disponemos los usuarios no realizan los pasos necesarios para que la calibración clásica de Kinect sea efectiva.

3.1.1. Fase de calibración

Definimos la calibración como el proceso necesario que debe realizar un usuario para que la aplicación pueda identificar cada una de las articulaciones del cuerpo. Para ello deben suceder una serie de acontecimientos.

1. La aplicación detecta uno o varios cuerpos, visualmente vemos por pantalla siluetas que corresponden a cada uno de los usuarios.

2. Una vez detectadas las siluetas, pasará a un modo de espera, que no cambiará hasta que el usuario pase a la posición de calibración (figura 4) o salga de la escena. Si sale de la escena y vuelve a entrar repite el proceso desde el punto 1.

3. En el momento que el usuario se coloca en la postura PSI (se denomina PSI dada la similitud a la letra griega psi) debe realizar pequeños movimientos para acabar de ajustar la posición.

4. Cuando la posición es aceptada por la aplicación, empieza a seguir los movimientos de la persona, además visualmente muestra por pantalla unas líneas que representan el esqueleto de la persona, como se observa en la figura 3.1.

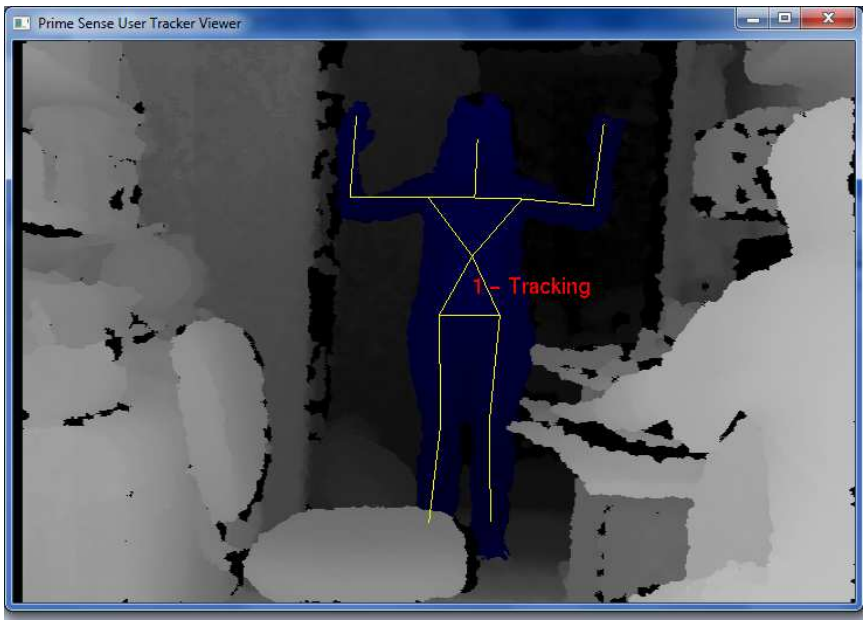


Figura 3.1: Postura de calibración PSI.

3.1.2. Método de calibración automática

Consideramos como calibración automática el hecho de poder pasar de los puntos 1 al 4 del apartado anterior sin pasar por el 2 y el 3, es decir conseguir seguir los movimientos de la persona sin tener que pasar por la postura de calibración PSI.

Esto lo conseguimos gracias a una de las últimas versiones de la librería OpenNI ya que incluye los métodos que muestra el cuadro 3.1 en la clase XnSkeletonCaptability. Estos métodos se encargaran de guardar y cargar la información necesaria para poder seguir a una persona sin pasar por la fase de calibración.

```
SaveCalibrationDataToFile(userID, nombreFichero);
LoadCalibrationDataToFile(userID, nombreFichero);
```

Cuadro 3.1: Funciones para guardar y cargar calibraciones desde un fichero.

Para aplicarlo al proyecto ha sido necesario guardar una serie de ficheros de calibración, el método para conseguirlos es mediante la calibración clásica comentada en el apartado anterior y añadiendo un último paso que consiste en la llamada al método de guardar la calibración en un fichero.

En caso contrario es necesario llamar al método de cargar la calibración y esperar unos segundos para que se reajusten las características de seguimiento al nuevo usuario. Hay que tener en cuenta que la información guardada en el fichero no siempre es apropiada para cualquier usuario, por esa razón es necesario tener varios fichero de calibración preparados.

Es conveniente no usar este método siempre que sea posible, ya que aunque podamos utilizar la información de calibración entre diferentes usuarios siempre será mejor utilizar la propia.

3.2. Esqueleto

El esqueleto es el resultado de la aplicación de una calibración sobre un usuario ya sea automática o mediante el sistema clásico. Nos muestra información sobre la posición actual de la persona y será el principal recurso de datos para reconocer las acciones.

Como información relevante para la aplicación, se extrae información de 15 puntos del cuerpo.

- Cabeza.
- Cuello.
- Hombro izquierdo/derecho.
- Codo izquierdo/derecho.
- Mano izquierda/derecha.
- Torso (centro de masa).
- Cadera izquierda/derecha.
- Rodilla izquierda/derecha.
- Pie izquierdo/derecho.

Extraemos la información utilizando un objeto del tipo XnSkeletonJointPosition el cual se usa como contenedor de la información de cada “joint”, utilizando el método GetSkeletonJointPosition de la clase XnSkeletonCaptability.

```
GetSkeletonJointPosition(userID,NombreArticulacion,jointDest);
```

Cuadro 3.2: Función para obtener un joint.

Una vez obtenida la información del joint, podemos acceder a la información en coordenadas reales del joint. Se puede observar en el cuadro 3.3 un código de ejemplo para obtener las coordenadas de la cabeza del usuario 1.

```
XnSkeletonJointPosition joint;
g_UserGenerator.GetSkeletonCap().GetSkeletonJointPosition(
   1,XN_SKEL_HEAD,joint);
double x,y,z;
XnPoint3D pt = joint.position;
x = pt[0].X;
y = pt[0].Y;
z = pt[0].Z;
```

Cuadro 3.3: Código ejemplo, para obtener las coordenadas de la cabeza usuario 1.

Una vez definido el método para la extracción de datos, es necesario guardarla para ser tratada posteriormente. El sistema de almacenamiento consta de un documento de texto, en el que se guarda la información. Siguiendo el formato del cuadro 3.4, la primera línea esta formada por el número de usuario al que pertenece esta pose y el número de frame desde que se inicio la reproducción. El resto de líneas están formadas por las coordenadas reales de cada uno de los puntos. Además el último campo hace referencia a la veracidad del punto, que solo puede tomar dos valores, 1 o 0, y quiere decir si el punto está medido realmente o si se trata de una interpolación teniendo en cuenta la información de los puntos anteriores.

```
IdUsuario   N °Frame
Xcabeza      Ycabeza      Zcabeza       Ecabeza
Xcuello      Ycuello      Zcuello       Ecuello
XhomzIzq     YhomzIzq     ZhomzIzq      EhomzIzq
XhomzDer     YhomzDer     ZhomzDer      EZhomzDer
XcodoIzq     YcodoIzq     ZcodoIzq      EcodoIzq
XcodoDer     YcodoDer     ZcodoDer      EcodoDer
XmanoIzq     YmanoIzq     ZmanoIzq      EmanoIzq
XmanoDer     YmanoDer     ZmanoDer      EmanoDer
XcaderaIzq   YcaderaIzq   ZcaderaIzq    EcaderaIzq
XcaderaDer   YcaderaDer   ZcaderaDer    EcaderaDer
XrodillaIzq YrodillaIzq ZrodillaIzq  ErodillaIzq
XrodillaDer YrodillaDer ZrodillaDer  ErodillaDer
XpieIzq      YpieIzq      ZpieIzq       EpieIzq
XpieDer      YpieDer      ZpieDer       EpieDer
Xtorso       Ytorso       Ztorso        Etorso
```

Cuadro 3.4: Formato archivo de puntos.

3.3. Descripción de comportamientos

Describimos las acciones que queremos reconocer como un conjunto ordenado de posturas. Dichas posturas se definen por las coordenadas reales de 15 puntos en un modelo 3D correspondientes a los puntos indicados en el apartado anterior.

La cantidad de posturas que definen un gesto viene dada por la velocidad de muestreo del dispositivo (15 ó 30 fps) y la duración del gesto:

$$Posturas = velocidadMuestreo \times duracionGesto$$

Cuadro 3.5: Formula calculo de postura para una acción.

Antes de poder reconocer una acción es necesario conocer en detalle dicha acción. Para ello bastaría con realizar una vez dicha acción y guardar cada una de las poses. El inconveniente es que

una acción quedaría acotada de forma estricta a como el usuario modelo ha realizado la acción. Para solucionar esto opta por repetir varias veces el mismo gesto para realizar una media y poder generalizar la acción evitando detalles puntuales de cada repetición.

En este punto teniendo una acción con la media no seria suficiente dado que cada persona es diferente y puede variar en la manera de realizar una acción. Por tanto la media debe estar formada por varios usuarios modelo. Este método aun no es del todo correcto ya que para no todas las personas tienen el mismo tamaño ni las mismas proporciones, este problema se soluciona aplicando una normalización.

3.3.1. Normalización de Poses

El proceso de normalización para las poses consiste en transformar la información sobre las coordenadas de cada punto de manera que:

- No influya la posición de la persona en el escenario (no importa cuan cerca o lejos este de la cámara).
- No influya el tamaño de la persona.

Para evitar el primer punto hacemos una traslación de coordenadas de manera que el punto (0,0,0) de la postura sea el punto que inicialmente refiere al cuello. Se decide elegir el cuello como punto de partida ya que es el punto más estático en condiciones normales. El método para conseguir esto consiste en restar a cada punto las coordenadas del punto del cuello. (e.g y así para cada uno de los puntos). Al referir todos los puntos desde un punto propio eliminamos la información de posicionamiento en el escenario.

En referencia al tamaño de la persona optamos por aplicar un factor de escala asumiendo que las proporciones humanas son similares. La manera escogida para conseguirlo consiste en calcular el ancho máximo de la pose e imponer que sea de una medida determinada. Esto lo conseguimos dividiendo el ancho máximo entre la medida escogida. Al resultado lo denominaremos factor de escala. Posteriormente se multiplica cada coordenada de cada punto por este factor de escala de manera que todas las poses pasan a tener el mismo ancho y las mismas proporciones respecto a ese ancho.

$$AnchoMaximo \div MedidaDeterminada = FactorEscala$$

$$P_{Joint}(x,y,z) = P_{Joint}(x,y,z) \times FactorEscala$$

Cuadro 3.6: Formula factor de escala.

Aplicando los pasos anteriores no aseguramos que dos personas diferentes, en una misma pose en diferentes puntos del escenario, lleguen a tener las mismas coordenadas una vez normalizado, pero se consiguen asemejar lo suficiente como para ser comparables.

El resultado de aplicar esta normalización es una serie de 42 valores. De los 45 valores originales se eliminan 3 de ellos (los referentes al cuello) ya que al aplicar la normalización, éstos quedan con valor (0,0,0) y no aportan nada a la aplicación.

3.3.2. Acciones para comparar

Una vez normalizadas las poses tenemos que definir el conjunto de poses que formarán la acción. Anteriormente se comenta que es necesario disponer de un conjunto de acciones (la misma) realizadas repetidas veces por diferentes usuarios. Para definir de forma final la acción se aplica el siguiente método.

- Una vez normalizadas las poses, se calcula la media de poses que tiene cada acción (hay que tener en cuenta que no siempre se tarda lo mismo en realizar una misma acción.
- Colocamos los 42 valores de cada pose de la acción en una matriz como muestra el cuadro 3.7:

$$\begin{pmatrix} Joint1X & Joint1Y & \cdots & Joint14Z \\ \vdots & \vdots & \ddots & \vdots \\ Joint1X & Joint1Y & \cdots & Joint14Z \\ Joint1X & Joint1Y & \cdots & Joint14Z \end{pmatrix} \begin{matrix} Pose1 \\ \vdots \\ PoseN-1 \\ PoseN \end{matrix}$$

Cuadro 3.7: Posicionamiento de poses en la matriz.

- Una vez la matriz esta completa se redimensiona cambiando el numero de filas ajustándolo a la media, para ello se utiliza la función de OpenCV.

```
CvResize(matrizOrigen, matrizDestino, CV_INTER_LINEAR));
```

Cuadro 3.8: Función para redimensionar una matriz.

- Cuando todas las acciones poseen el mismo tamaño se procede a calcular la media aritmética para cada coordenada de cada joint de cada pose.

El resultado de toda esta normalización es lo que se utiliza en el algoritmo DTW para localizar un gesto dentro de una secuencia.

3.4. Reconocimiento con DTW

El algoritmo de reconocimiento de patrones Dynamic Time Warping (DTW) nos permite ver el grado de similitud entre dos señales teniendo en cuenta que su duración puede ser variable. En la figura 7 observamos dos señales similares con variaciones en el tiempo y su correspondiente asociación temporal.

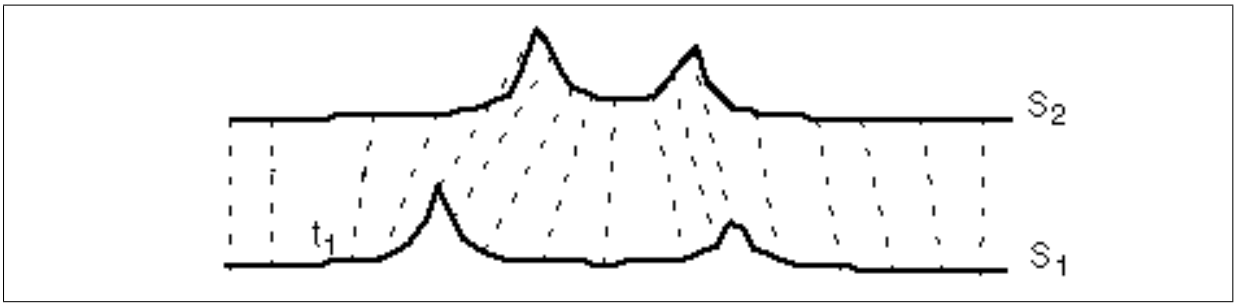


Figura 3.2: Dos señales similares

El algoritmo daría como buena la similitud entre las dos señales ya que ambas tienen el mismo comportamiento y únicamente varían en el tiempo. Este hecho es fácilmente aplicable al reconocimiento de acciones dado que la variación en el momento de realizar una acción es pequeña en cuanto al movimiento en sí, pero es muy diferente en cuestiones de velocidad, e.g la acción de saludar con la mano es muy similar para cualquier usuario, pero la velocidad en la que se realiza la acción es muy variable en función de la persona.

3.4.1. Funcionamiento DTW

Para el funcionamiento del algoritmo se necesitan 3 parámetros:

- El patrón que se está buscando (Qué se busca).
- Un flujo de datos de entrada (Dónde se busca).
- Un parámetro numérico de corte (Cuándo se acepta como similar).

Una vez definidos los parámetros se trabaja con una matriz que debe tener tantas filas como vectores de datos tenga el patrón y tantas columnas como se desee. Cada columna corresponderá a un vector de datos de entrada. En el cuadro 3.9se muestra la distribución de estos parámetros.

	DatoEntrada1	DatoEntrada2	DatoEntrada3	DatoEntrada4
Dato1Patron	a_{11}	a_{12}	a_{13}	a_{14}
Dato2Patron	a_{21}	a_{22}	a_{23}	a_{24}
...	...	...	...	...
DatoNPatron	a_{n1}	a_{n2}	a_{n3}	a_{n4}

Cuadro 3.9: Distribución del patrón y datos de entrada.

Una vez vista la distribución de los datos, es necesario rellenar la matriz, cada valor de la matriz se calcula de la siguiente manera:

- Se calcula la distancia euclidiana[17] entre los correspondientes datos (fila-columna).
- Al valor calculado, se le suma el mínimo de los tres valores colindantes calculados hasta el momento.

Ejemplo de cálculo del valor a_{22}:

$$d_{22} = \sqrt{\sum(Dato2Patron_i - DatoEntrada2_i)^2}$$

$$a_{22} = d_{22} + Min(a_{11}, a_{12}, a_{21})$$

Cuadro 3.10: Ejemplo calculo, valor a_{22}.

Hay que tener en cuenta que en las posiciones de la fila y la columna uno no se puede aplicar directamente estas fórmulas ya que se saldrían de la matriz, esto se soluciona aplicando unos valores como muestra el cuadro 3.11.

		DatoEntrada1	DatoEntrada2	DatoEntrada3	DatoEntrada4
	0	0	0	0	0
Dato1Patron	∞	a_{11}	a_{12}	a_{13}	a_{14}
Dato2Patron	∞	a_{21}	a_{22}	a_{23}	a_{24}
...	∞	...	...	...	...
DatoNPatron	∞	a_{n1}	a_{n2}	a_{n3}	a_{n4}

Cuadro 3.11: Solución para limites de matriz.

Podemos observar en el cuadro 3.12 una implementación en pseudocódigo del algoritmo DTW.

```
int DTWDistance(char s[1..n], char t[1..m]) {
  declare int DTW[0..n, 0..m]
  declare int i, j, cost
  for i := 1 to m
     DTW[0, i] := infinity
  for i := 1 to n
     DTW[i, 0] := infinity
  DTW[0, 0] := 0
  for i := 1 to n
     for j := 1 to m
          cost:= d(s[i], t[j])
          DTW[i, j] := cost + minimum(DTW[i-1, j  ],     // insertion
  return DTW[n, m]
}
```

Cuadro 3.12: Pseudocódigo del Algoritmo DTW.

Visto como se calcula cada valor de la matriz, falta ver como se decide si las datos de entrada son similares al patrón. Para eso entra en juego el parámetro anteriormente definido como de corte. Para este paso es necearia la comparación con los valores de la última fila.

Si alguna de las comparaciones es cierta, entonces podemos afirmar que se ha encontrado en los datos de entrada una señal lo suficientemente parecida como para aceptarla como igual.

Para encontrar el punto de origen (donde empieza la señal) hay que recorrer la matriz desde el punto en que la comparación fue cierta, eligiendo el mínimo de entre los tres valores superior, izquierdo y esquina izquierda superior hasta llegar a la fila 1, como muestra el ejemplo del cuadro 3.13 (suponiendo que el valor de corte es 5.0).

		DatoEntrada1	DatoEntrada2	DatoEntrada3	DatoEntrada4
	0	0	0	0	0
Dato1Patron	∞	1.010	4.351	9.164	12.323
Dato2Patron	∞	5.110	2.033	7.419	16.119
Dato3Patron	∞	13.909	7.118	3.028	7.061
Dato4Patron	∞	5.892	15.562	6.895	4.024

Cuadro 3.13: Recorrido inverso una vez encontrada la secuencia.

3.4.2. Aplicación al proyecto

La aplicación al proyecto es directa, se identifica cada parámetro de la siguiente forma:

- Patrón: Es cualquier acción que esté definida tal y como indica el punto 3.3.
- Entrada: El flujo de datos que se pretenda analizar, estos datos pueden ser obtenidos de una grabación realizada con Kinect o extraídos directamente de la reproducción de Kinect.
- Valor de Corte: El valor de corte se describe en el apartado 4.2.

CAPÍTULO 4

Resultados

Para observar el funcionamiento de la aplicación se diseñan dos experimentos, uno sobre una base de datos propia y otro sobre una aplicación real (niños con TDAH).

4.1. Datos

En cada uno de los experimentos se trabaja con un tipo de datos descritos en los siguientes apartados.

4.1.1. Datos experimento 1

Los datos del experimento forman parte de una base de datos que contiene lo siguiente:

- Por un lado tenemos un directorio denominado Datos_Kinect, que se divide en varios subdirectorios del tipo PersonaX donde X es el identificador de la persona que participa en el experimento. Dentro del directorio existen otros cinco subdirectorios con nombre gestoY donde Y es el identificador del gesto que se analiza. Finalmente dentro de cada directorio de gesto existen tres ficheros con extensión *.dat que contienen la información obtenida con la cámara Kinect.
- Por otro lado está el directorio Gestos que contiene cinco subdirectorios con nombre GestoX donde X es el identificador del gesto, que contiene 100 ficheros con extensión *.dat, que contienen la información de las acciones realizadas por los usuarios (un fichero para cada repetición).

El fichero points.dat contiene la información de los puntos tal y como se extraen de la fuente de datos (cámara o vídeo). Contiene bloques de 16 líneas que a su vez contienen los puntos en coordenadas reales, el usuario al que pertenecen y el identificador de frame como en el cuadro 3.4.

El fichero userFrames.dat contiene la cantidad de frames que hay en el fichero points.dat para cada usuario. El formato correspondiente se muestra en el cuadro 4.1.

```
Usuraio1 CantidadFrames
Usuario2 CantidadFrames
...
```

Cuadro 4.1: Formato fichero userFrames.dat.

El fichero Usr1_NcE.dat contiene la información normalizada de las poses existentes en el fichero points.dat para el usuario 1. Para facilitar el trabajo, los datos de los ficheros points.dat para esta base de datos están fijados siempre en el usuario 1. Como separador entre una acción y las repeticiones se utilizan dos filas de ceros, como muestra el cuadro 4.2.

```
0.000000 0.000000 0.000000 ... 0.000000 0.000000 0.000000
dato1    dato2    dato3    ... dato40   dato41   dato42
dato1    dato2    dato3    ... dato40   dato41   dato42
 ...      ...      ...      ...  ...      ...      ...
dato1    dato2    dato3    ... dato40   dato41   dato42
dato1    dato2    dato3    ... dato40   dato41   dato42
0.000000 0.000000 0.000000 ... 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 ... 0.000000 0.000000 0.000000
dato1    dato2    dato3    ... dato40   dato41   dato42
dato1    dato2    dato3    ... dato40   dato41   dato42
 ...      ...      ...      ...  ...      ...      ...
dato1    dato2    dato3    ... dato40   dato41   dato42
dato1    dato2    dato3    ... dato40   dato41   dato42
0.000000 0.000000 0.000000 ... 0.000000 0.000000 0.000000
```

Cuadro 4.2: Formato fichero Usr1_NcE.dat.

Dentro de los directorios GestoX del directorio Gestos los ficheros N.dat donde N es un número de 1 a 100, contienen la información de los gestos normalizados de los usuarios (un fichero por acción).

4.1.2. Datos experimento 2

En este experimento se utiliza como fuente de datos los puntos extraídos de los esqueletos obtenidos en los diferente vídeos de los niños con TDAH. Los datos consisten en la filmación durante 2 horas en entorno escolar de 3 chicos diagnosticados con TDAH. Como patrón a buscar se preparan unas acciones de muestra, imitando el gesto que queremos encontrar en los datos originales, estas muestras corresponden al movimiento de agachar la cabeza sobre la mesa, el gesto parte de una posición en la que el usuario está sentado en una silla completamente recto, con los brazos pegados al cuerpo y las piernas perpendiculares al suelo, la acción termina cuando el usuario inclina la cabeza y el cuerpo sobre la mesa sin mover los brazos ni las piernas.

4.2. Parámetro de corte DTW

El parámetro de corte no se puede estimar a simple vista para el cálculo de los valores, empleamos el siguiente método:

1. Se obtiene un grupo de acciones conocidas, es decir se sabe cuantas acciones hay y de que tipo son.
2. Se pone como patrón la acción sobre la cual queremos calcular el valor de corte y como parámetro de entrada cada una de las acciones del conjunto anterior.
3. Mediante un bucle, se hace pasar cada acción por el algoritmo y se anota en que casos se acepta la acción como similar. Al final de cada iteración se calcula un valor (cuadro 4.3) que evalúa la efectividad del parámetro de corte. El bucle se inicia con un valor bajo y se incrementa cuanto se desee.

$$efectividad = \frac{casosFavorablesAccionN}{totalAccionesN + casosDesfavorablesAccionN}$$

Cuadro 4.3: Formula para evaluar la efectividad del parámetro de corte.

La formula coge valores entre 0 y 1, siendo 0 el peor de los cosos y 1 el mejor, cada una de la variables representa:

- casosFavorableAccionN: Cuantas de las acciones de tipo N que pasan posan por el algoritmo dan positivo como resultado.
- totalAccionesN: Cantidad original de acciones que hay del tipo N.
- casosDesfavorablesAccionN: Cantidad de acciones que dan positivo como resultado y no son del tipo N.

Se elegirá como valor de corte el mínimo valor que resulte una efectividad más elevada. En el caso de que algún parámetro de corte de como resultado una efectividad de 1, no será necesario continuar con el bucle, ya que el resultado no puede ser mejor que 1 y el hecho de aumentar el valor de corte aumenta el riesgo de aumentar los casos desfavorables.

4.3. Validación

Como método de validación se utiliza el método conocido como “leave-one-out” que consiste en dejar fuera del conjunto de cálculo un dato de la muestra y luego se evalúa el dato excluido con el resultado del cálculo. Es necesario repetir el proceso de cálculo para cada uno de los datos de la muestra.

El resultado de la exclusión de un archivo se suma a los valores existentes en una matriz de confusión. Dicha matriz es una matriz de ceros cuadrada de dimensión igual al numero de acciones diferentes que existen en el conjunto. Los datos se distribuyen tal como muestra el cuadro 4.4.

$$\begin{pmatrix} ctdPositvos_{11} & ctdPositvos_{12} & \cdots & ctdPositvos_{1n} \\ ctdPositvos_{21} & ctdPositvos_{22} & \cdots & ctdPositvos_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ ctdPositvos_{n1} & ctdPositvos_{n2} & \cdots & ctdPositvos_{nn} \end{pmatrix}$$

Cuadro 4.4: Distribución de la Matriz de Confusión.

Los valores $ctdPositvos_{ij}$ hacen referencia a la cantidad de valores aceptados agrupados por tipo y gesto:

- i: hace referencia al gesto del patrón.
- j: hace referencia al grupo que pertenece la acción.

La efectividad de la validación se calcula evaluando las formulas del cuadro 4.5sobre la matriz de confusión.

$$efecctividadClasificacion = \frac{sumaElementosDiagonal}{sumaTotal}$$

$$efecctividadReal = \frac{sumaElementosDiagonal}{sumaElementosNoDiagonal + TotalAcciones}$$

Cuadro 4.5: Cálculo de la efectividad de la matriz de confusión.

La primera fórmula nos dice la efectividad de la clasificación, es decir, nos da un porcentaje de las acciones que ha clasificado correctamente, mientras que la segunda nos da un valor de efectividad real, ya que tiene en cuenta las acciones que no han sido clasificadas y las que se han resultado como falsos positivos.

El valor de las efectividades se comprende entre 0 y 1, de manera que 0 es el peor resultado y 1 el mejor.

En la aplicación se implementa siguiendo los siguientes pasos:

1. Se dispone de un conjunto de *N* acciones de varios tipos (se conoce exactamente cuantas acciones hay de cada tipo.
2. Se excluye la primera acción del gesto 1, y se realizan los cálculos de media de los gestos y los parámetros óptimos de corte para cada acción.
3. Se aplica el algoritmo DTW con el gesto excluido como parámetro de entrada, cada uno de las medias de las acciones como patrones a reconocer y los valores óptimos como parámetro de corte.
4. Se anota en la matriz de confusión el resultado de la evaluación del algoritmo.
5. Se repite el proceso desde el punto 2 un total de N veces, excluyendo cada vez una acción diferente.
6. Finalmente se calcula el valor de efectividad sobre la matriz de confusión.

4.4. Resultado Experimento 1

Para el experimento 1 se utiliza una base de datos como la definida en el apartado 4.1.1. La base de datos utilizada consta concretamente de 500 acciones realizadas por un grupo de 9 personas que realizan 5 tipos de acciones correspondientes a:

- Saludar con la mano derecha .
- Señalar al frente con la mano derecha.
- Aplaudir.
- Saltar.
- Agacharse.

Se pueden ver algunos ejemplos de las poses utilizadazas para cada acción (figuras 4.1,4.2,4.3,4.4 y 4.5).

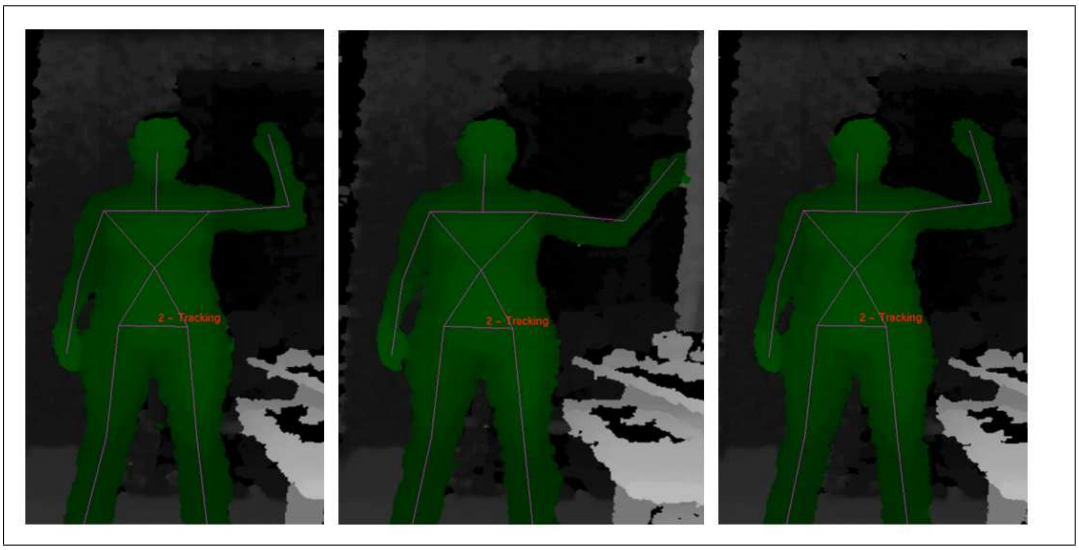


Figura 4.1: Muestra de poses para la acción de saludar con la mano derecha.

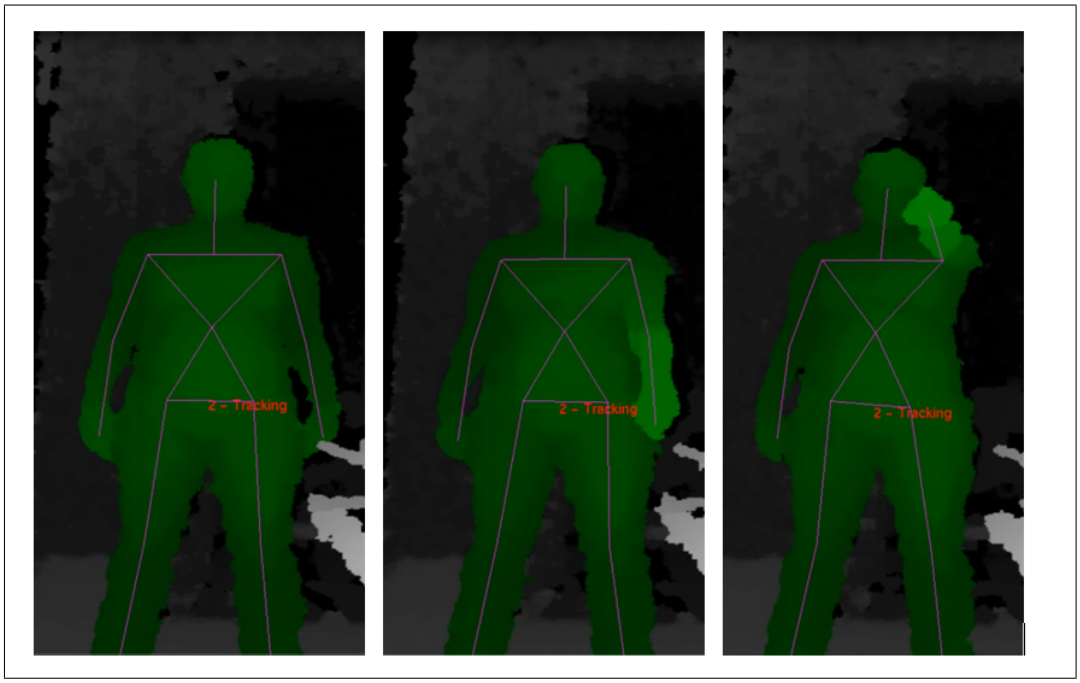


Figura 4.2: Muestra de poses para la acción de señalar al frente con la mano derecha.

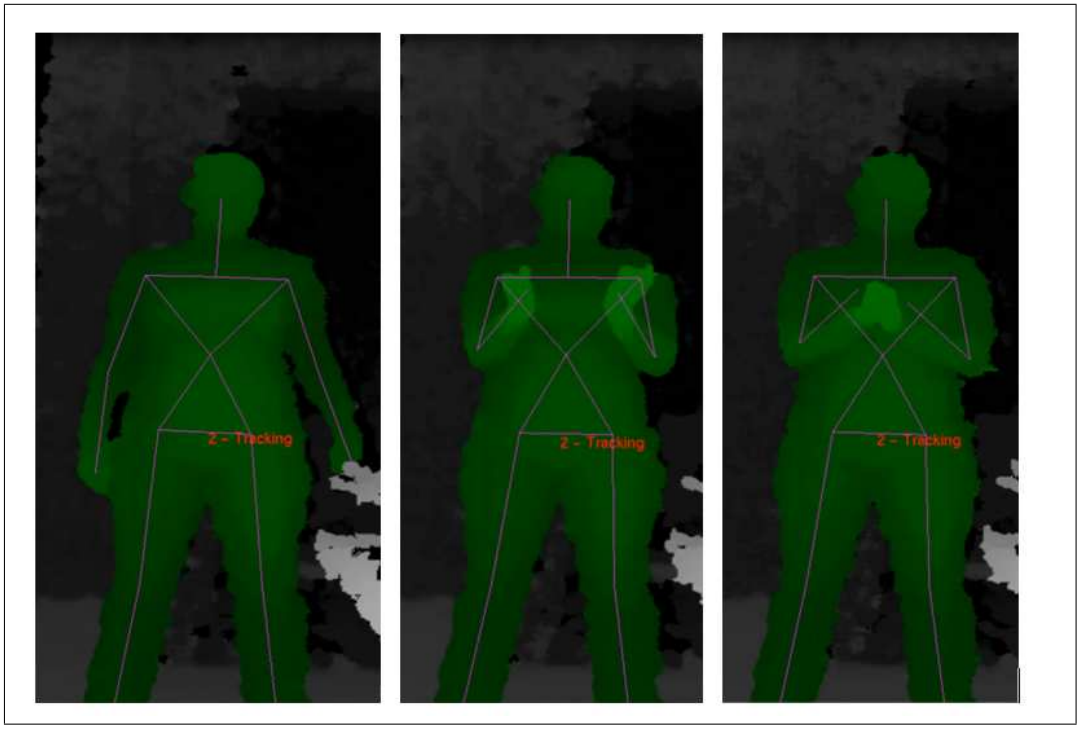


Figura 4.3: Muestra de poses para la acción de aplaudir.

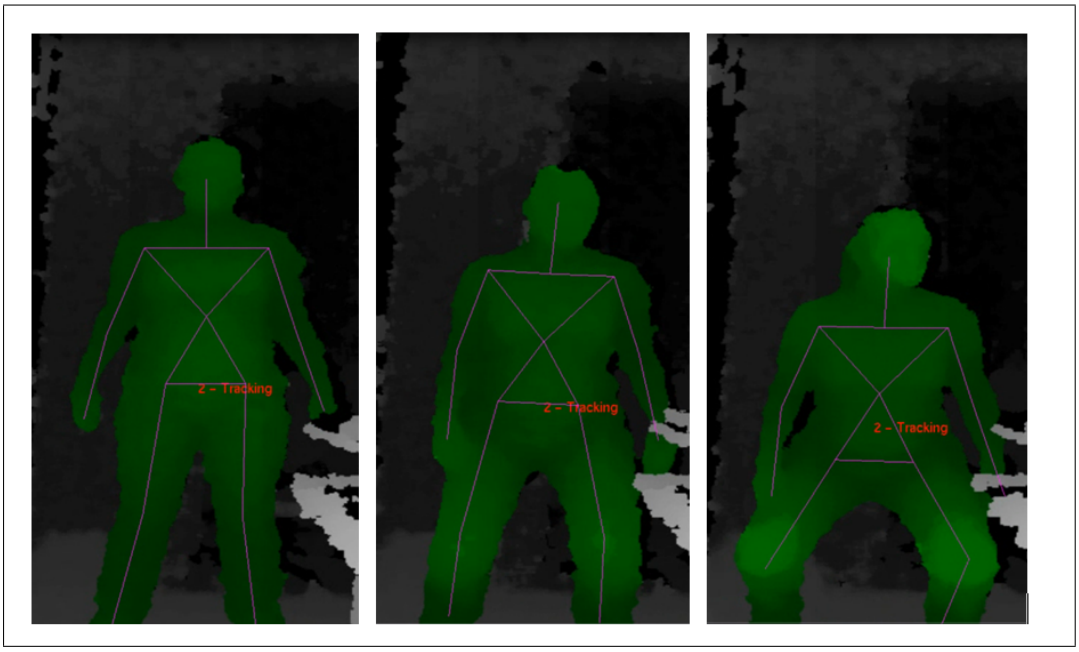


Figura 4.5: Muestra de poses para la acción de agacharse.

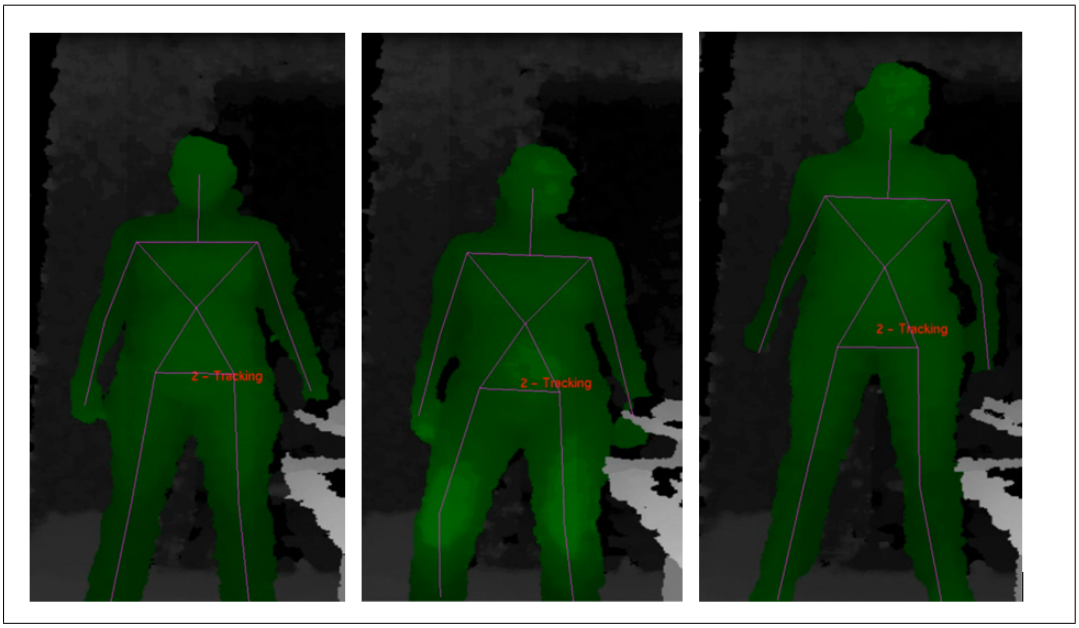


Figura 4.4: Muestra de poses para la acción de saltar.

Se lanza el algoritmo con un incremento de los valores de corte de 50. Después de la ejecución del algoritmo se obtiene como resultado unos valores de corte óptimos de :

- Gesto 1 - Saludar con la mano derecha $Vc = 1200$.

- Gesto 2 - Señalar al frente con la mano derecha. $Vc = 700$.
- Gesto 3 - Aplaudir $Vc = 1150$.
- Gesto 4 - Saltar $Vc = 450$.
- Gesto 5 - Agacharse $Vc = 600$.

y la matriz de confusión que muestra el cuadro 4.6.

$$MatrizConfusion = \begin{pmatrix} 99 & 0 & 0 & 0 & 0 \\ 0 & 86 & 0 & 15 & 0 \\ 0 & 5 & 95 & 6 & 0 \\ 0 & 0 & 0 & 13 & 0 \\ 0 & 0 & 0 & 7 & 57 \end{pmatrix}$$

Cuadro 4.6: Matriz de confusión experimento 1.

El cálculo de la efectividad se realiza mediante la fórmula del cuadro 4.5:

$$efectividadClasificacion = \frac{99+86+95+13+57}{99+86+95+13+57+15+5+6+7} = 0{,}91$$

$$efectividadReal = \frac{99+86+95+13+57}{500+15+5+6+7} = 0{,}66$$

Cuadro 4.7: Efectividad experimento 1

Analizando en detalle los resultados de la matriz de confusión se puede ver que:

- Gesto 1 - Clasificación excelente: 99 Casos favorables, 0 falsos positivos y 1 acción no clasificada. El gesto de saludar con la mano derecha esta bastante generalizado y no presenta problemas para ser clasificado.
- Gesto 2 - Clasificación media: 86 Casos favorables, 15 falsos positivos y 14 acciones no clasificadas. Señalar al frente presenta algunas diferencias significativas en función del usuario que realiza la acción. Durante la toma de datos se observó que es un gesto mas personal, algunos usuarios suben el brazo completamente estirado y otros flexionan el codo en diferente grados.

- Gesto 3 - Clasificación buena: 95 Casos favorables, 11 falsos positivos y 5 acciones no clasificadas. El gesto de aplaudir también esta bastante generalizado y no presenta demasiados problemas de clasificación.
- Gesto 4 - Clasificación mala: 13 Casos favorables, 0 falsos positivos y 87 acciones no clasificadas. Se observa tanto en la toma de datos como en el resultado que saltar es una acción completamente personal. Algunos usuarios se agachan más que otros y el comportamiento de los brazos para tomar impulso es bastante propio.
- Gesto 5 - Clasificación media: 57 Casos favorables, 7 falsos positivos y 43 acciones no clasificadas. En la acción de agacharse se observan las mismas características que en la acción saltar, pero los resultados son algo mejores.

El resultado de la efectividad de clasificación es bastante bueno, ya que tiene una efectividad del 91 %, aunque viendo el resultado de efectividad real (0.66) vemos que es del todo bueno dado que hay una cantidad importante de acciones que no son clasificadas (150).

4.5. Resultado Experimento 2

Como primer resultado del experimento se puede valorar el funcionamiento del proceso de calibración automática. En la mayoría de los casos se observa que el proceso de calibración no resulta efectivo, no por el proceso en sí, sino por el hecho de que los niños están bastante estáticos (sentados detrás de una mesa) el software no es capaz de detectar que hay un cuerpo, al no detectar la presencia de cuerpos no es posible aplicar la calibración. En los casos en los que se puede aplicar hay secciones en los que los datos no son válidos para el experimento ya que muchos de los puntos que se extraen no son válidos (el software interpola la posición donde cree que está el punto). Por todo lo anteriormente comentado, se reduce mucho el volumen de información del que se disponía inicialmente.

Encontramos un ejemplo en el que aunque nos permite aplicar la calibración automática, esta falla debido a que no identifica bien a los usuarios. Define un usuario a base de mezclar tres (dos niños y el profesor) eso hace que el reconocimiento de las articulaciones falle y las identifique de manera equivocada (figura 4.6).

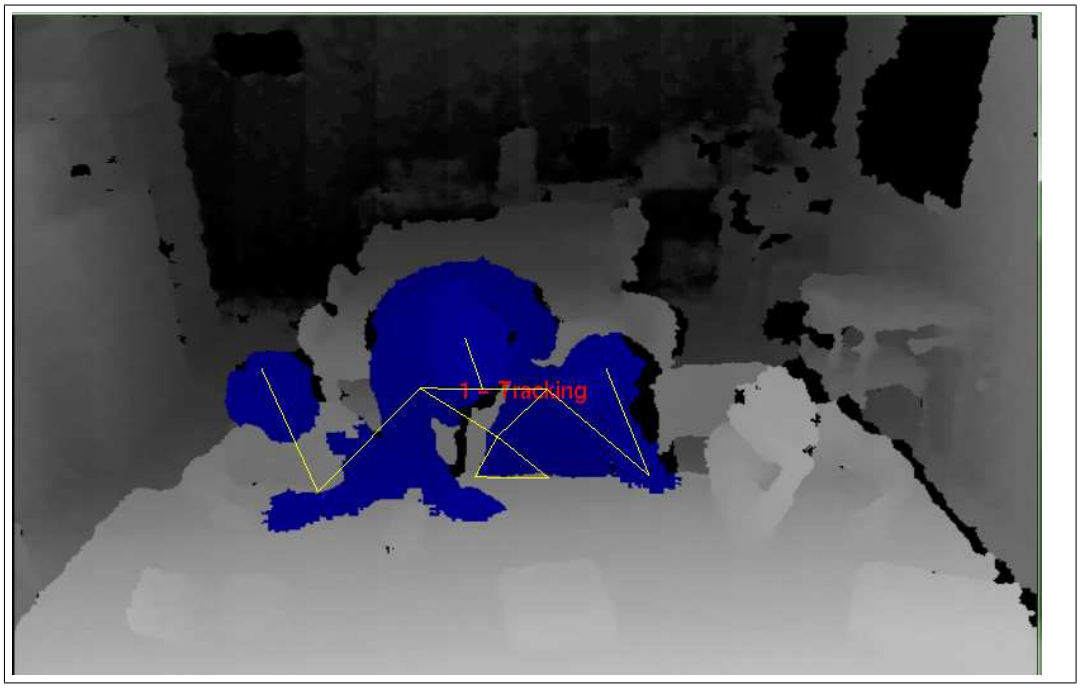


Figura 4.6: Ejemplo de calibración incorrecta.

En la figura 4.7 se muestra un frame de una de las grabaciones realizada a los niños con TDAH durante una clase de matemáticas, se puede observar el resultado de aplicar la calibración automática a la grabación en la figura 4.8.

Figura 4.7: Imagen de muestra en RGB de los niños con TDAH en clase de matemáticas.

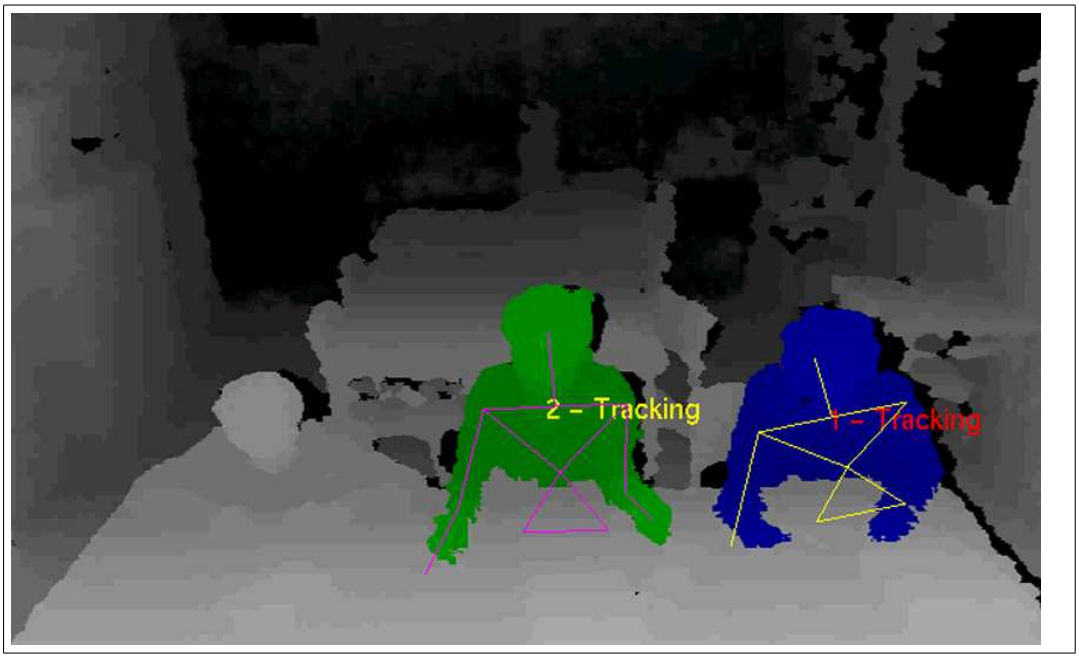


Figura 4.8: Resultado de la aplicación de la calibración automatica sobre el video de la figura 4.7.

En este caso la calibración resulta correcta en los usuario 1 y 2, existe un problema con el usuario 3 ya que permanece bastante estático durante toda la secuencia y el software no es capaz de identificar un cuerpo en esa posición.

En la figura 4.9 se muestra un frame de una de las grabaciones realizada a los niños con TDAH durante una sesión de trabajo con el ordenador, se puede observar el resultado de aplicar la calibración automática a la grabación en la figura 4.10.

Figura 4.9: Imagen de muestra en RGB de los niños con TDAH en una sesión de trabajo con un ordenador.

Figura 4.10: Resultado de la aplicación de la calibración automatica sobre el video de la figura 4.9.

En este caso ninguno de los usuarios es detectado por el software, por tanto no es posible aplicar la calibración sobre ellos.

En el segmento de información que la extracción de datos funciona de forma más correcta se busca el patrón definido en el apartado 4.1.2, como resultado de la búsqueda obtenemos:

- Da como falso positivo el inicio de la secuencia, los primero 67 frames.
- Se encuentra un reconocimiento en los frames comprendidos entre [199-206] el inicio del gesto y en [249-256] el final del gesto. El parámetro de corte se eleva hasta 6000 que en comparación con el experimento 1 es mucho mas elevado, esto es debido a la pérdida de puntos por oclusiones.

En la figura 4.11 y 4.12 se muestra la secuencia donde se encuentra el patrón que se esta buscando.

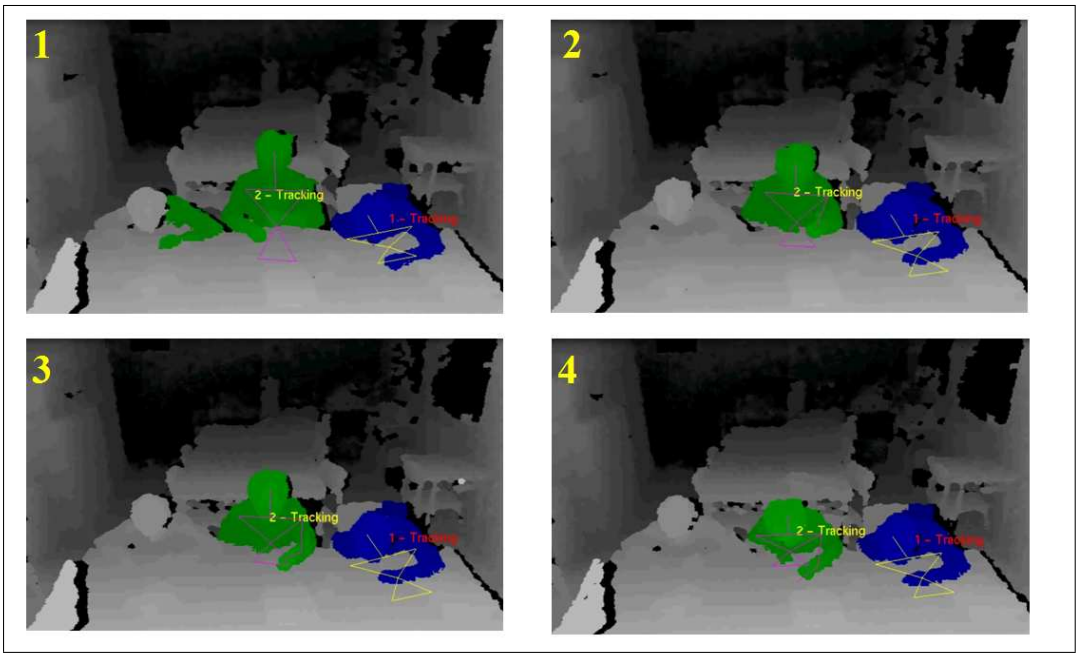


Figura 4.11: Secuencia con el patrón encontrado. Parte 1

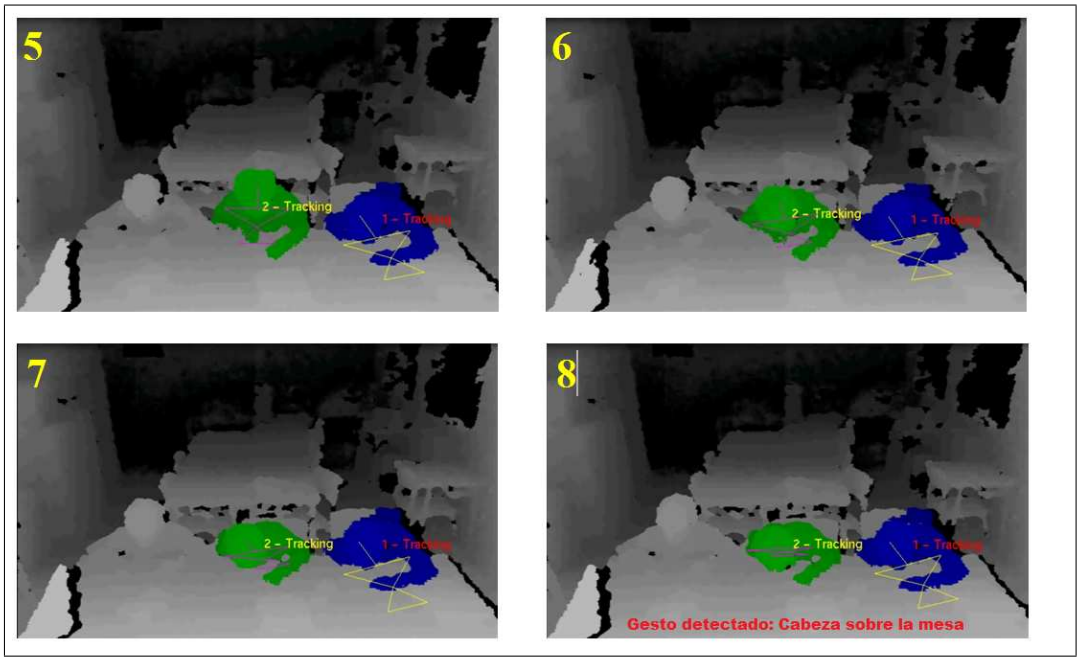


Figura 4.12: Secuencia con el patrón encontrado. Parte 2

CAPÍTULO 5

Conclusiones y trabajo futuro

En primer lugar se puede evaluar el resultado de la calibración automática. Se observa por un lado que es una gran ventaja el hecho de poder evitar la fase de calibración clásica. Aunque se ha demostrado que funciona, se observa que es necesario que exista una cantidad de movimiento importante por parte del usuario al que se quiere calibrar para que esta funcione correctamente. Por tanto, siempre que sea posible es recomendable utilizar la calibración clásica. Sólo se utilizará la calibración automática en casos en los que los datos ya estén tomados sin haber realizado la fase de calibración o en casos en los que la detección de la persona es buena (no hay oclusiones, esta de frente,etc.).

En cuanto al algoritmo DTW, se observa que funciona correctamente siempre que en la secuencia donde estamos buscando exista un tramo muy similar al patrón que se esta buscando y en función al parámetro de corte asociado. Este parámetro de corte tiene el inconveniente de que no puede fijarse a simple vista. Es necesaria una fase de validación como la implementada o ajustar por tanteo en secuencias de datos en las que se sabe a priori que existe el patrón que se busca.

En la búsqueda de gestos la similitud de lo que se busca afecta de forma muy directa y pueden darse casos donde un gesto que es claramente el mismo que otro el algoritmo no los clasifique como iguales, como por ejemplo la acción de saludar puede no ser reconocida como igual por el hecho de tener las piernas en una posición significativamente diferente a las de la media que se utiliza como patrón.

Se plantea una mejora del algoritmo DTW para solucionar el problema del párrafo anterior, el planteamiento de mejora consiste en aplicar una serie de pesos a cada una de las articulaciones que participan en la acción de manera que influyan en mayor o menor medida en función de lo que aporten al gesto. Ppor ejemplo, en el gesto de saludar con la mano derecha, se tendría que dar mas importancia al movimiento de la mano derecha y el codo derecho. Con un grado un poco menor de importancia se tendría que contar la mano y el codo izquierdo, para así diferenciarlo de otros gestos. Finalmente se aplicaría un peso prácticamente nulo a las piernas ya que no aportan casi nada al gesto de saludar con la mano derecha.

CAPÍTULO A

Contenido del CD

En el CD adjuntado con la memoria, se puede encontrar:

- /Memoria.pdf : Archivo con una copia de la memoria en formato PDF.
- /InstalarOpenNI: Directorio con los instaladores del driver para Kinect y el framework OpenNI.
- /Codigo: Directorio con el código de las aplicaciones tanto de la aplicación para extraer información de Kinect así, como de la aplicación del análisis de los datos.
- /Ejecutable: Directorio con los archivos ejecutables de las diferentes aplicaciones.

CAPÍTULO B

Instalación OpenNI

Para un correcto funcionamiento de la librería OpenNI es necesario seguir los siguiente pasos:

1. En el directorio del CD /InstalarOpenNI, ejecutar y seguir las instrucciones de instalación del fichero OpenNI-Win32-1.1.0.38-Dev.msi.
2. En el directorio del CD /InstalarOpenNI, ejecutar y seguir las instrucciones de instalación del fichero SensorKinect-Win32-5.0.0.exe.
3. En el directorio del CD /InstalarOpenNI, ejecutar y seguir las instrucciones de instalación del fichero NITE-Win32-1.3.1.3-Redist.msi.

En el punto 3 es necesaria una contraseña, que está disponible en el archivo Contraseña_NITE.txt del directorio /InstalarOpenNI del CD.

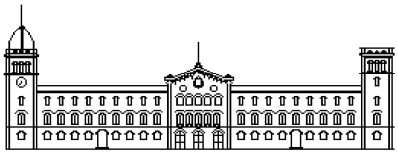
Índice de figuras

Índice de cuadros

Bibliografía

- [1]Información OpenCV: http://www.willowgarage.com/pages/software/opencv
- [2]Bradski, Gary & Kaehler, Adrian, ”Learning OpenCV: Computer Vision with the OpenCV Library”, O’Reilly Media 2008.
- [3]http://en.wikipedia.org/wiki/Hidden_Markov_model
- [4]Kimmel G, Shamir R, ”A block-free hidden Markov model for genotypes and its application to disease association”, MEDLINE, 2005.
- [5]http://en.wikipedia.org/wiki/Dynamic_time_warping
- [6]Wang K & Gasser T, ”Alignment of Curves by Dynamic Time Warping”, JSTOR 2008.
- [7]http://www.tovalive.com/processament-imatges/control-pc-amb-webcam-la-ma-com-a-mouse/
- [8]www.tovalive.com/processament-imatges/reconeixement-de-gestos-%E2%80%93-numeros-amb-les-mans
- [9]https://github.com/OpenNI/SampleAppSinbad
- [10]http://kinecthacks.net/aquila-icub-teleoperation
- [11]http://kinecthacks.net/3d-drawing-with-the-kinect/
- [12]http://www.primesense.com/
- [13]http://www.openni.org/
- [14]http://openkinect.org/wiki/Main_Page
- [15]http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/

- [16]http://www.primesense.com/?p=515
- [17]http://es.wikipedia.org/wiki/Distancia_euclidiana



ENGINYERIA TÈCNICA EN INFORMÀTICA DE SISTEMES

UNIVERSITAT DE BARCELONA

Treball fi de carrera presentat el dia de de 2011
a la Facultat de Matemàtiques de la Universitat de Barcelona,
amb el següent tribunal:

Dr. President

Dr. Vocal 1

Dr. Secretari

Amb la qualificació de: