Universidad del País Vasco
Euskal Herriko Unibertsitatea

Facultad de Informática

Grado de Ingeniería Informática

▪ Proyecto Fin de Grado ▪

Ingeniería de Computadores

Sistema de comercio electrónico para el sector textil: generar maniquíes 3D a partir de imágenes del sensor Kinect

Ibon Olabarria Iñarra

Junio 2014

informatika fakultatea
facultad de informática

Sistema de comercio electrónico para el sector textil: generar maniquíes 3D a partir de imágenes del sensor Kinect

Resumen

Este proyecto consiste en analizar la viabilidad técnica un sistema de información para mejorar el comercio electrónico de compra –venta de ropa. El nuevo sistema de comercio electrónico ayudará al usuario a escoger el tamaño de la prenda con más precisión. El sistema consiste en calcular las medidas esenciales del cuerpo del cliente utilizando el dispositivo Kinect. Posteriormente, estas medidas se introducen en el programa Makehuman que genera un maniquí en 3D. Con la idea de que en el futuro se puede utilizar el maniquí para probar virtualmente prendas de vestir, se ha ideado un método, utilizando el programa Blender, que genera ropa en 3D con fotografías de prendas. Se generan dos ejemplos (pantalón y camiseta) para ilustrarlo.

Palabras clave: Makehuman, Kinect, Blender

Ehungintzarako komertzio-sistema elektronikoa: 3D manikien sorrera Kinect sensorearen irudietatik abiatuta

Laburpena

Proiektu hau merkataritza sistema elektroniko bidezko arropa salerosketak hobetzeko bideragarritasun teknikoa aztertzean datza. Merkataritza sistema elektroniko berri honek jantzien neurri egokiena aukeratzen lagunduko ditu erabiltzaileak. Sistemak gorputzaren oinarrizko neurriak hartuko ditu Kinect gailuari esker. Ondoren, neurri hauek Makehuman programa informatikoan sartzen dira 3Dko manikia eratuz. Etorkizunean manikia birtualki eratutako arropak janztea da helburua, bide berri bat irekiz Blender programaren bidez zeinek jantziaren argazkietatik abiatuz, 3Dko modeloa garatzen duena. Metodoa erakusteko bi adibide sortzen dira, praka eta kamiseta.

Hitz gakoak: Makehuman, Kinect, Blender

Electronic commerce system for the textile sector: generating 3D mannequins from Kinect sensor images

Abstract

This project consists of analyzing the technical feasibility of an information system to improve the buying and selling clothing through Internet. The new system helps the user to choose the right size of the garment. In order to develop this system we first calculate the essential measurements of the client´s body using the Kinect device. Subsequently, these measurements are introduced in a program called Makehuman which will generate a 3D mannequin. Having in mind that in the near future the mannequin could be dressed virtually, we have also developed a method using Blender to generate 3D clothing from pictures. We finally have illustrated two examples (jeans and T-shirt) of how Blender works.

Key words: Makehuman, Kinect, Blender

0. Introducción

Este proyecto de fin de grado se desarrolla durante el curso 2013-2014. La motivación del mismo es la mejora del comercio electrónico de compra-venta de ropa por Internet. El problema a tratar es el siguiente: una vez adquirida la prenda por Internet, si no resulta adecuada a nuestra talla, nos vemos obligados a devolverla. Esto supone al *cliente* consumo de tiempo además de un coste económico (debido a las tasas de envío) y a la *empresa* que provee la ropa, la posible pérdida de la fidelización del cliente.

El comercio electrónico también conocido como ***e-commerce (electronic commerce)*** ha crecido notablemente en los últimos tiempos. Cada vez son más las empresas que apuestan por esta forma de negocio como puede verse en el siguiente gráfico.

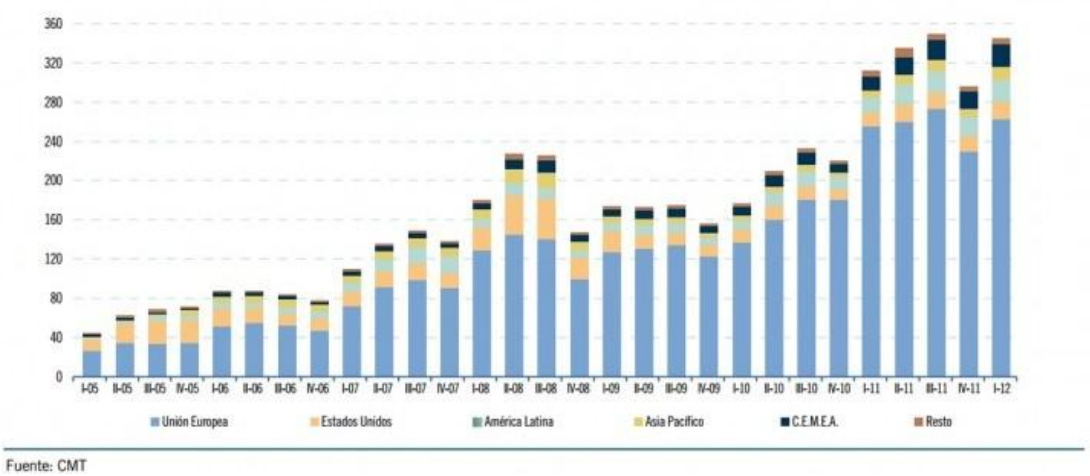


Figura 1: evolución trimestral del volumen de negocio del comercio electrónico desde el exterior con España por aéreas geográficas (millones de euros). (*)

(*)Fuente: http://www.usc.es/

Normalmente cuando compramos una prenda por internet, podemos elegir la talla y el color. Pero las mismas tallas en diferentes prendas suelen tener distintas medidas y suele ser difícil conocer las equivalencias entre las tallas estándar de diferentes países. Las prendas de vestir se realizan utilizando patrones de costura (se especificará más adelante este término). La hipótesis principal de este trabajo es que de poder ajustarse las prendas sobre un maniquí personalizado del cliente podrían mejorar de forma radical el comercio online de ropa:

- Solo disponiendo de las medidas del cliente y de la ropa, se pueden seleccionar las prendas que se ajustan al cliente explicando en qué zonas la prenda es grande o pequeña evitando devoluciones y potenciado la venta de complementos (por ejemplo cinturones).
- Si se dispone de una modelo 3D realista, lo que es en realidad un maniquí de costura y de información básica sobre el tipo de patrón (por ejemplo la clase pantalón campana), se puede producir un video personalizado en el que la persona pueda ver de forma realista cómo le queda la prenda y se pueden gestionar online los arreglos derivando a un taller de costura la prenda.
- Si se dispone del patrón exacto de las prendas se puede automatizar su ajuste a un cuerpo determinado en una cortadora industrial: las fábricas textiles podrían así elaborar prendas por encargo.

- Se está abordando automatización del proceso de generación del maniquí a partir de imágenes de cámaras fotográficas (2D) o estereográficas (3D), categoría esta última en la que incluimos el sensor Kinect de la consola de juegos Xbox 360. Con esta tecnología se podrían elaborar de patrones de costura para las personas enfermas o dependientes con movilidad limitada, creando prendas que incrementen la ergonomía y faciliten la a veces dolorosa tarea ponerse o ponerle la ropa

Con este proyecto se quiere conseguir la mejora del comercio electrónico de compra-venta de ropa y se apuesta por investigar el problema más difícil: cómo automatizar la extracción del maniquí mediante un sensor Kinect.. La idea es crear una empresa en la que sea posible capturar por medio de imágenes las medidas del cliente y en esta memoria se analiza cómo se puede hacer esto utilizando al dispositivo Kinect. Con ello se pretende que la ropa tenga las medidas adecuadas para que el cliente evite los problemas de tallaje mejorando sustancialmente la fidelización de los clientes con la empresa comercial o industrial.

Las prendas son creadas mediante patrones, los cuales establecen las medidas de la ropa y las restricciones para su arreglo. Uno de los objetivos de este trabajo es analizar cómo se puede conseguir que la prenda diseñada se adecue a las medidas del cliente. De esta manera es posible crear *ropa personalizada*.

En definitiva, nos gustaría crear una empresa de comercio electrónico que posea una base de datos con las medidas de los clientes necesarias para levantar un modelo 3D realista de su cuerpo, con la posibilidad de crear ropa personalizada que se pueda probar y mostrar sobre el maniquí.

Como ya se ha mencionado, el tipo de clientes que más podría beneficiarse son las personas con problemas de movilidad, que suelen tener dificultades a la hora de comprar ropa. Muchos de ellos no tienen posibilidad de probársela y se ven obligados a adquirir ropa antiestética y de mayor tamaño del que necesitan. Con esta empresa se quiere conseguir solucionar esta clase de problemas: gracias al dispositivo Kinect se tomarían las medidas de los interesados y seleccionar o generar los patrones más adecuados. Con los patrones, se podría crear ropa fácilmente y probarla sobre el maniquí dotado de la misma movilidad que el cliente. Por ejemplo, se podrían ofrecer pantalones que en los laterales tengan costuras de *velcro* y ciertos anclajes a la silla a personas en silla de ruedas, para las que es muy costoso ponerse unos pantalones al uso. Los patrones podrían modificarse para que los usuarios puedan tener pantalones adecuados a su minusvalía e incluso unas instrucciones de cómo colocárselos sin ayuda utilizando presillas, cintas, cremalleras y otros elementos auxiliares de costura.

Mi objeto de estudio tiene múltiples opciones debido al número creciente de aplicaciones de la informática al mundo de la moda, al modelado corporal, etc.:

- Con el dispositivo Kinect se pueden crear maniquíes con las medidas de las personas.
- En el campo como la biometría, el estudio de métodos automáticos para el reconocimiento único de humanos basados en rasgos físicos, mejoraría identificando a las personas gracias a sus medidas corporales.

1. Objetivo y campo de aplicación

El objetivo de este proyecto es realizar el análisis de viabilidad de un sistema de información capaz de (i) crear ropa en 3D y (ii) crear maniquís con las medidas del usuario con el fin personalizar la recomendación y confección de prendas de vestir.

Para este proyecto se pretende recoger y estudiar documentos y referencias bibliográficas en los que expliquen cómo realizar las dos tareas antes mencionadas, así como estudiar las herramientas de software actualmente disponibles de forma gratuita que se necesitarían para tal fin. La ropa en tres dimensiones se pretende diseñar mediante las imágenes delantera y trasera de una prenda real. De este modo, tendrá un color más real.

Kinect es un dispositivo que capta imágenes en color y por medio de un sensor de infrarrojos es capaz de obtener imágenes de profundidad. Estas imágenes o mapas pretenden mostrar al usuario la profundidad de la imagen; es decir, indican a qué distancia se encuentran los objetos que capta la cámara.

Este proyecto utilizará programas en código abierto como Blender para dibujar ropa en tres dimensiones y Makehuman para crear maniquíes en 3D con las medidas del usuario. Para tomar las medidas se utilizará el dispositivo Kinect: he investigado varias formas de desarrollar aplicaciones para medir el cuerpo humano. Utilizaré Blender porque es una herramienta muy útil para crear objetos en tres dimensiones. Con todo ello se pretende abrir puertas para futuras aplicaciones en este campo.

El desarrollo de aplicaciones utilizando Kinect está muy extendido. Existen comunidades como OpenKinect [1] en la que personas interesadas dan a conocer sus aplicaciones realizadas con la cámara Kinect.

Para la creación de ropa en 3D existen también muchos tipos de software. Debido a que Blender es un programa de código abierto, está programado en ***Python*** y puede ser desarrollado por los usuarios, este es un programa muy completo.

2. Normas para consulta

En esta memoria se aplica la norma española UNE 157801 “Criterios generales para la elaboración de proyectos de Sistemas de Información” [2], a la que se ciñe el índice de la memoria.

3. Términos y definiciones

Para la finalidad de este proyecto se aplican los siguientes términos y definiciones:

Kinect:

Controlador de juego de la videoconsola Xbox360, desarrollado por Microsoft. Acrónimo de *Kinetic Connect*.

Makehuman:

Aplicación de gráficos de software libre para la creación de maniquíes en tres dimensiones.

Blender:

Programa informático multi-plataforma con el código abierto para funcionalidades como el modelado, texturizado, iluminación, animación y *renderizado* de gráficos en tres dimensiones.

MatLab

Herramienta multi-plataforma de software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio (lenguaje M)

Python

Lenguaje de programación funcional en el que están basados Makehuman y Bender.

Patrón de costura

Plantilla realizada sobre papel para ser copiada en el tejido y fabricar una prenda de vestir. Esta plantilla está compuesta por las piezas de la prenda que habrá que cortar, armar y coser.

GIMP

Programa libre y gratuito de edición de imágenes con funcionalidades similares al programa de pago *Photoshop*.

Gliffy

Servicio en línea (disponible en *Chrome Web Store*) que permite crear diagramas y compartirlos con otras personas. Realiza diagramas de clases, diagrama UML, planos de casa, mapas conceptuales…

Pixel

Un píxel (en inglés, *picture element*) es la menor unidad homogénea en color que forma parte de una imagen digital

Renderizar

Es el proceso de crear una imagen (o video) mediante el cálculo de la iluminación partiendo de una imagen en tres dimensiones.

4. Requisitos generales de la documentación del proyecto

4.1. Titulo

Sistema de comercio electrónico para el sector textil: generar maniquíes 3D a partir de imágenes.

4.2. Documentación

El proyecto consta de los siguientes documentos:

- Índice general
- Memoria
- Anexos
    - A: Fase 1: Estudio de Makehuman
    - B: Fase2: Creación deprendas en Blender
    - C: Fase 3: Calcular medidas a partir de Kinect
    - D: Fase 4: Estudio de base de datos de usuarios y prendas
    - E: Cargar maniquí con las medidas de usuario
    - F: Blender
    - G: Detectar el dispositivo Kinect en MatLab
    - H: Installation de Kinect for Windows en MatLab
    - I: Kinect
- Presupuesto orientativo
- Gestión del tiempo del proyecto
- Conclusiones y futuras aplicaciones
- Bibliografía
- Enlaces de interés
- Seguimiento y control

5. Índice general

6. Índice de imágenes

7. Índice de tablas

8. Índice de algoritmos (código)

9. Memoria

9.1. Introducción

Actualmente para la compra-venta de ropa por Internet, la gente la elige basándose únicamente en la talla de la prenda que desea comprar. El objetivo de principal de este proyecto es desarrollar un sistema de captura de medidas del usuario y crear ropa en tres dimensiones con imágenes reales.

9.2. Objetivo

El primer objetivo del proyecto es generar una aplicación que sea capaz de capturar 19 medidas corporales: longitudes y anchuras, ayudándonos del dispositivo Kinect. Este dispositivo posee herramientas que facilitarán la extracción de las medidas. Posteriormente, estas medidas se trasladan al programa de software libre de creación de maniquíes llamado Makehuman. Una vez finalizada, el cliente podrá visualizar un maniquí con sus medidas corporales.

El segundo objetivo del proyecto es investigar la creación de ropa en tres dimensiones. Para ello se utiliza el programa de software libre Blender, que permite crear ropa para los maniquíes y añadir imágenes de ropa real, proporcionando a la prenda una apariencia real.

9.3. Antecedentes

En la compra-venta de ropa electrónica no existe la posibilidad de que el usuario la escoja totalmente adecuada a su tamaño. Tiene que elegir el tamaño de la prenda en base a su talla. Actualmente la ropa se crea utilizando patrones de costura. La tela se corta utilizando dichos patrones como guía. Es el diseñador de la ropa el que crea estos patrones y son las empresas las que se dedican a confeccionar las prendas. Estas se basan en estos patrones para confeccionarlas.

Hoy en día las medidas costura se toman a mano. Las medidas que se toman son las siguientes:

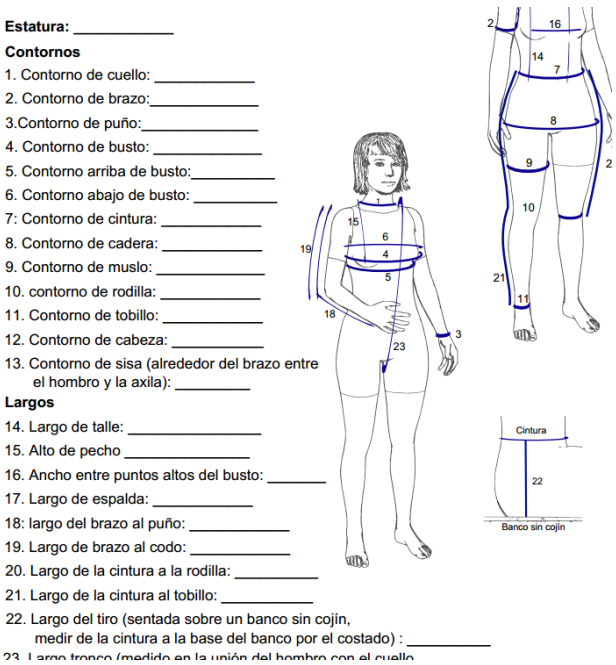


Figura 2: Medidas de costura. (*)

(*) Fuente: http://anilegra.blogspot.com.es/

Para las mediciones se utilizan cinta métricas flexibles como la de la figura 3:

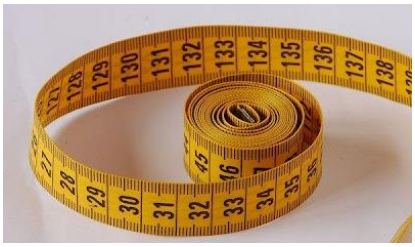


Figura 3: Cinta métrica. (*)

(*) Fuente: http://www.lessentiel.lu/

El programa que se va a utilizar para crear maniquíes (Makehuman) tiene la posibilidad de vestir al usuario con ropa en tres dimensiones: consiste en una malla 3D única, elaborada por un conjunto de artistas junto con un conjunto procedimientos para deformarla según la medida, edad y tono muscular del cuerpo que se quiera crear. Las prendas de las que dispone el programa consisten en una malla 3D predefinida y se elaboran con el *plugin* de Blender *MakeClothes* cortando y deformando la malla del cuerpo humano. Queremos investigar si se puede crear ropa cambiando la textura de la prenda con fotografías de forma que tengan un aspecto real.

9.4. Descripción de la situación actual

El principal problema de comprar ropa en Internet es no poder probar las prendas en las que estamos interesados. Para solucionar este problema existen ***vestidores o probadores virtuales*** ***[4]***. Funcionan como un espejo, permitiendo al usuario ver como lo sienta la ropa. Estos programas posibilitan cambiar las prendas, los colores y otras características agilizando la compra. Estos programas colocan la ropa al usuario.

Para que el usuario pueda visualizar la ropa en su cuerpo, la prenda se superpone sobre imagen del usuario. El tamaño de la prenda siempre es el mismo, lo único que hace es encoger y aumentar el tamaño dependiendo de si cubre el cuerpo del usuario adecuadamente.

Figura 4: probador virtual *El armario de la tele. (*)*

(*) Fuente: http://www.elarmariodelatele.com/

Suelen añadirse complementos tales como cambiar el fondo y el entorno de la imagen. Por ejemplo, si estamos comprando ropa de verano, en vez de que aparezca el fondo de la tienda, podemos vestirnos virtualmente en la playa o si estamos comprando ropa de invierno, podemos vestirnos en un entorno

nevado. Otro componente que se está incorporando últimamente es la posibilidad de compartir las imágenes en las redes sociales.

Figura 5: ejemplo de probador virtual. (*)

(*) Fuente: http://retail-innovation.com/

Existen empresas que se dedican a desarrollar probadores virtuales.

	Modificación características de ropa	Optimizado para móviles	Mediciones corporales	Introducir foto de cara	Ajustar medidas de la ropa al usuario
Probador virtual del Corte Inglés [5]	Si (color…)	si	Si (peso, altura, forma del cuerpo, tamaño pecho, color piel)	si	no
365 look [6]	Si	No	Si (las predicen según edad, peso, altura y forma del cuerpo)	no	no
Zugara [7]	si	si	Si(las predicen según edad, peso, altura y forma del cuerpo)	No	no
Fitnect [8]	Si	si	Si	No	no
FITS [9]	si	si	Si (las predicen según edad, peso, altura y forma del cuerpo)	si	no

Tabla 1: comparativa entre distintas web de probadores virtuales.

La característica en común que tienen estos probadores virtuales es que la ropa se ajusta al usuario. Es decir, La prenda en su totalidad aumenta o reduce su tamaño hasta cubrir al usuario. Nosotros queremos conseguir:

- Capturar las medidas de costura del usuario, que es el objetivo principal de este proyecto
- Utilizando esas medidas de costura, modificar toda la prenda para que se ajuste a la persona. Esto en último término se hará mediante patrones de costura, pero en este

proyecto solo se aborda cómo ajustar la fotografía en plano de la ropa sobre una prenda ya modelada

9.5. Métodos, herramientas, modelos, métricas y prototipos

En este proyecto se aborda el análisis de la viabilidad del sistema de información descrito elaborando prototipos que demuestren que es posible tomar las medidas de un cliente a través de un sensor Kinect, levantar un maniquí virtual en Makehuman a partir de las mismas y estudiar si es posible mapear las fotografías de una prenda tomadas en plano por delante y detrás sobre una prenda ya construida con MakeClothes

Para este propósito es más adecuado utilizar una metodología ágil, basada en la interacción con el cliente mediante la entrega de código y documentos, como es *Scrum*.

9.5.1. Mecanismos de control de calidad aplicados

Para realizar los controles de calidad, se van mostrando los entregables al cliente, precisando de su visto bueno para continuar.

9.5.2. Referencias

Para la comprensión de la capacidad del dispositivo Kinect, se han observado varios trabajos de investigación aplicados a este campo. [10] [11] [12]

9.5.3. Requisitos iniciales

Se definen los siguientes requisitos para el proyecto:

- Análisis de mejora del sistema de comercio electrónico de compra venta de ropa ya realizado en el apartado de introducción.
- Aplicación que capture 19 medidas de costura del usuario.
- Método para crear prendas en tres dimensiones.

9.5.4. Alcance

Estos son los entregables del proyecto que obtendremos al final del proyecto.

0. *Documento con el diseño del sistema de comercio electrónico:*
   Breve resumen explicativo del objetivo principal de la empresa que se quiere crear.
1. *Documento con las modificaciones que hay que realizar en Makehuman:*
   Cómo introducir las medidas a mano y cómo obtener el maniquí con las medidas del usuario.
2. *Documento de explicación del proceso de crear ropa en Blender:*
   Pasos dados para crear ropa con imágenes de prenda reales.
3. *Aplicación para el cálculo de medidas a partir de Kinect:*
   Obtención de las medidas corporales del usuario.
4. *Base de datos con las medidas del usuario:*
   Medidas capturadas por Kinect para futuras aplicaciones.
5. *Memoria:*
   La memoria del proyecto.

9.5.5. Hipótesis y restricciones

Al inicio del proyecto se estimó que la parte de creación de ropa constituiría la tarea principal a realizar. Se pensó que sería sencillo dada la gran cantidad de información existente en la red. Respecto a

la captura de medidas con Kinect, al no haber tanta información se pensó que constituiría una tarea secundaria.

Sin embargo, al realizar la tarea de crear ropa no se fue progresando adecuadamente debido a las limitaciones que tiene el software libre. Hemos utilizado el sistema operativo Windows (condición técnica que no podíamos cambiar sobre la marcha) y hemos constatado que tanto MakeClothes como Blender presentan, al menos en este sistema operativo, una tremenda incompatibilidad entre las diferentes versiones además de la incompatibilidad (perdida de funcionalidad sin mensajes de error) entre Blender y los drivers de imagen de Intel que obligó en una máquina a sustituir el driver por el estándar de Windows. Por otra parte, las mallas de la ropa creada con MakeClothes y del cuerpo humano de Makehuman se corresponden punto a punto, así que también existe incompatibilidad con las diferentes versiones de Makehuman que progresivamente han ido eliminando triángulos de la estructura para convertirlos en cuadriláteros.

La estimación de tiempo inicial de esta fase del proyecto se fue agotando. Se decidió finalizar esta fase y dedicarse primero a la fase de medición corporal. Con la adquisición de conocimientos, se fueron cumpliendo los objetivos establecidos en la tarea finalizándose satisfactoriamente. La tarea de crear la ropa se sustituyó por la tarea más sencilla de mapear la textura de las prendas sobre las mallas ya creadas con MakeClothes que proporciona Makehuman.

9.5.6. Estudio de alternativas y viabilidad

La elección de software respecto a la tarea de crear ropa fue clara: Blender era el programa óptimo. Respecto a la captura de medidas, al no tener conocimientos se empezó a trabajar con el programa que venía por defecto con Kinect for wiindows SDK llamado *SkeletalViewer Walkthrough*, desarrollado en C++. Este programa muestra el uso de la interfaz la API natural del usuario (Natural User Interface, NUI) en Kinect For Windows SDK. Pero al no obtener los resultados esperados, se decidió cambiar al programa MatLab debido a su gran funcionalidad en el campo de las matemáticas.

9.5.7. Análisis de riesgos

Todo proyecto es susceptible de sufrir contratiempos no previstos durante su desarrollo. Para evitar que su impacto no altere el estado del proyecto, es importante prevenirlos y establecer soluciones.

9.5.7.1. Riesgos tácticos

Perdida de documentación

- Descripción: Al realizar un proyecto con gran cantidad de información, es posible que la información se pierda.
- Solución: Realizar copias de seguridad de la documentación, en carpetas online o locales.
- Impacto y probabilidad: Tendría un impacto crítico, pero la probabilidad que suceda es baja.
- Fases: Todas

Baja por enfermedad

- Descripción: Es probable que alguno de los miembros del proyecto enferme.
- Solución: ninguna.
- Impacto y probabilidad: impacto bajo ya que dos personas dirigen el proyecto y otras dos lo asesoran, probabilidad baja de que ocurra.

Fallo de comunicación

- Descripción: Los miembros pueden tener un fallo de comunicación y pueden producirse fallos de distinta magnitud.
- Solución: Utilizar distintos medios de comunicación: e-mail, teléfono,…
- Impacto y probabilidad: impacto crítico, probabilidad baja de que ocurra.

9.5.7.2. Riesgos formativos

Inexperiencia de tecnologías.

- Descripción: No tener suficientes conocimientos de las tecnologías o los lenguajes usados durante el proyecto.
- Solución: Dedicar tiempo para formación en las tecnologías y lenguajes desconocidos.
- Impacto y probabilidad: impacto moderado, probabilidad alta de que ocurra.

9.5.8. Organización y gestión del proyecto

9.5.8.1. Estimación de los objetivos

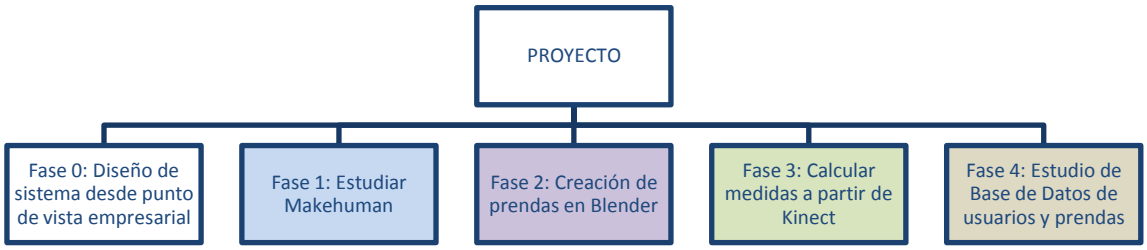


Figura 6: objetivos del proyecto

9.5.8.2. Estimación temporal

Fase	Nombre tarea	Horas estimadas
0	Diseño del sistema desde el punto de vista empresarial	5
1—makehuman	Estudiar de Python	5
	Análisis de Makehuman	25
	Documentar localización datos a modificar	15
	Reuniones	2
	Seguimiento y control	3
Horas totales		**50**
2—Blender	Estudio de Blender	10
	Crear ropa con Blender	35
	Documentar como crear ropa con Blender para Makehuman	20
	Reuniones	2
	Seguimiento y control	3
Horas totales		**70**
3—kinect	Documentar como introducir a mano las medidas en Makehuman	5
	Análisis del funcionamiento de Kinect y MatLab	35

	Calculo de medidas corporales obtenidas mediante Kinect	90
	Pruebas	10
	Reuniones	5
	Seguimiento y control	5
Horas totales		**150**
4—Base de datos	Análisis de requisitos de la aplicación web	15
	Estudio de creación base de datos	10
	Reuniones	5
	Seguimiento y control	5
Horas totales		**35**
Total proyecto		**310**

Tabla 2: estimación temporal.

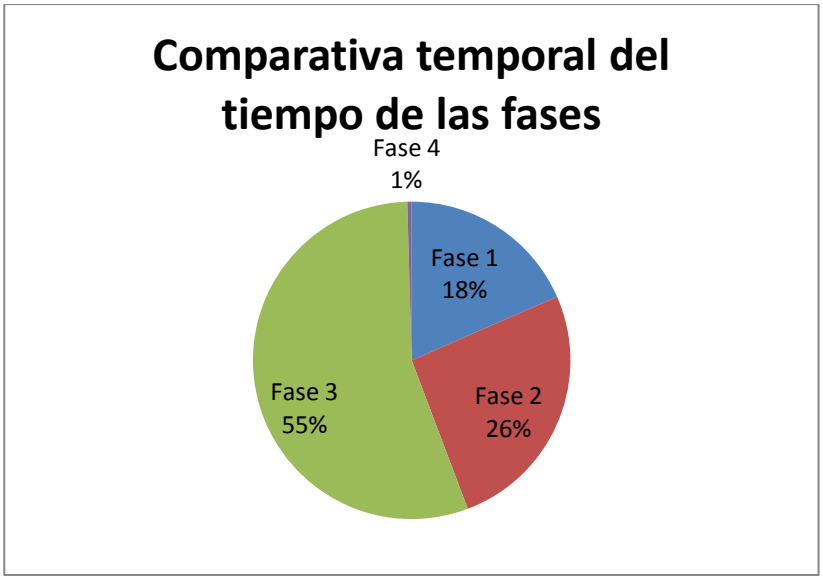


Figura 7: comparativa temporal de las fases.

9.5.8.3. Entregables de los objetivos:

Fase 0:

Entregable: Diseño del esquema del sistema de compra ventas

Fase 1:

Entregable: Documento con las modificaciones que hay que realizar en Makehuman

Fase 2:

Entregable: Colección de prendas básicas creadas con Blender y la memoria correspondiente

Fase 3:

Entregable: documento sobre cómo introducir las medidas en Makehuman y aplicación de cálculo de las medidas captadas por Kinect.

Fase 4:

Entregable: Estudio de Base de datos de usuarios y prendas.

9.5.8.4. Estructura de descomposición del trabajo (EDT)

A continuación se mostrará el diagrama de descomposición del trabajo.

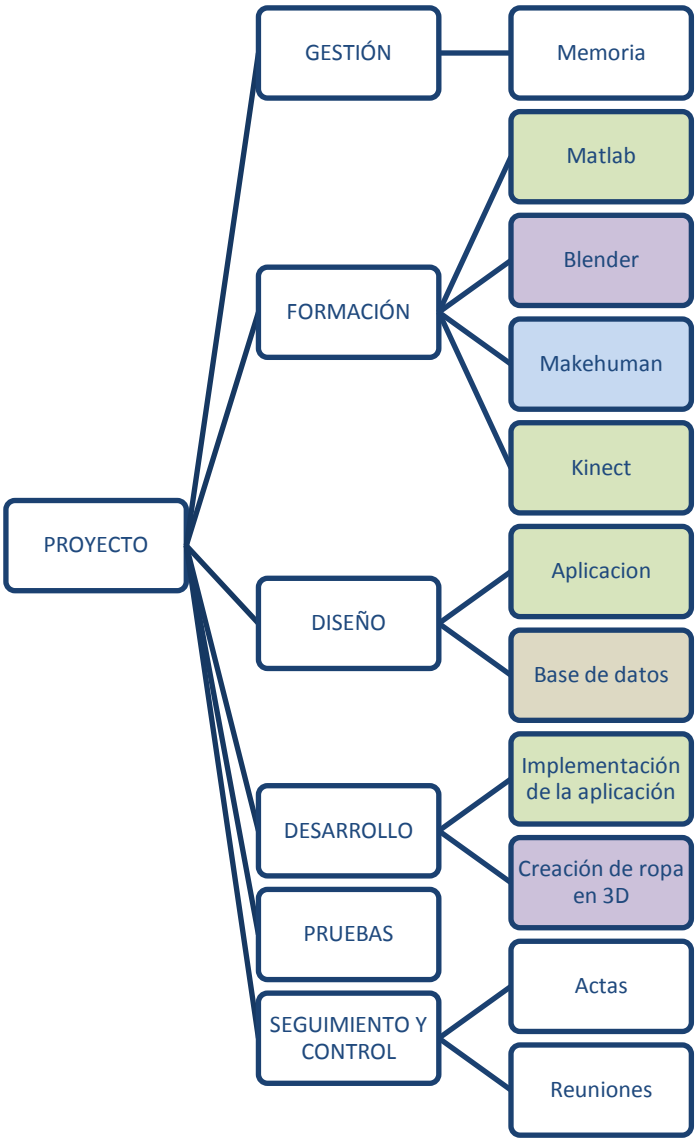


Figura 8: diagrama EDT

9.5.8.5. Listado de trabajo

Gestión:

La memoria se ocupa de esta parte del proyecto.

Formación:

Se estudia la utilización de diferentes programas y dispositivos. Los programas en cuestión son MatLab, Makehuman y Blender y el dispositivo será Kinect para la Xbox360.

Diseño:

Se diseña la aplicación para captar medidas de costura del usuario y un método para crear prendas en 3D.

Desarrollo:

Utilizado el diseño creado previamente se desarrollan la aplicación y el método.

Pruebas:

Se realizan pruebas para la constatación de la calidad.

Seguimiento y control:

Se realizan reuniones tanto con el cliente como con la directora del proyecto.

9.5.8.6. Descripción de las responsabilidades

Descripción de cada una de las responsabilidades que cumple cada uno de los integrantes del Proyecto de Fin de Grado:

- Proyecto Informático:
  - Ibon Olabarria.
    - Responsabilidad:
      - ✓ Director
      - ✓ Analista
      - ✓ Programador
      - ✓ Pruebas
- Proyecto académico:
  - Blanca Cases:
    - Responsabilidad:
      - ✓ Codirectora
  - Abdelmalik Moujahid
    - Responsabilidad
      - ✓ Codirector
  - Fadhi Dornaika
    - Responsabilidad:
      - ✓ Cliente
      - ✓ Asesor
  - Alireza Bosaghzadeh
    - Responsabilidad
      - ✓ Asesor

9.5.8.7. Comunicación

Analizadas las necesidades de comunicación entre los miembros del proyecto, se ha establecido los siguientes medios de comunicación:

- **Correo electrónico**: La mayor parte de la comunicación entre los miembros del proyecto se realiza mediante el uso del correo electrónico proporcionado por la universidad:

Miembro del proyecto	e-mail
Ibon Olabarria	iolabarria003@ikasle.ehu.es
Blanca Cases	blancarosa.cases@ehu.es
Fadhi Dornaika	fadi.dornaika@ehu.es
Abdelmalik Moujahid	abdelmalik.moujahid@ehu.es
Alireza Bosaghzadeh	alireza.bosaghzadeh@ehu.es

Tabla 3: miembros del proyecto.

- **Métodos de distribución de la información**: se ha creado una carpeta compartida en *Dropbox* para compartir archivos así como la edición de los mismos.

10. ANEXOS

10.1. ANEXO A: Fase 1: Estudio de Makehuman

10.1.1. FORMACIÓN

Makehuman es un programa en constante evolución. Debido a que la base de datos y el código están liberados, desde el primer prototipo que salió a la luz en el año 1999, ha habido 8 versiones del programa. Esto significa que las mallas de los maniquíes por ejemplo, han pasado de mallas con algunos triángulos hasta ser mallas poligonales libres de triángulos basadas completamente en pequeños cuadriláteros.

.

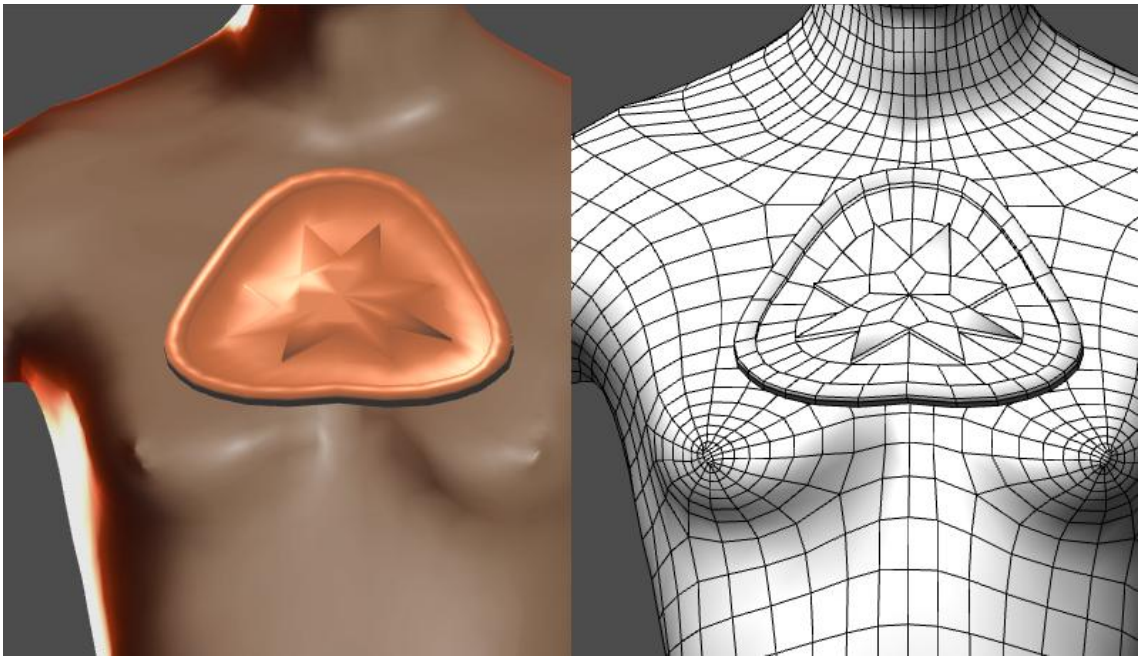


Figura 9: Ejemplo de malla utilizada por Makehuman. (*)

(*) Fuente: http://www.makehuman.org/

En este proyecto vamos a utilizar la séptima versión llamada *Makehuman 1.0 alpha 7*. Makehuman es un programa que permite que el modelo inicial que aparece en pantalla pueda ser modificado al gusto del usuario. Podemos cambiar características como el género, cambiar su estructura, anchura, edad, rasgos étnicos, la ropa a vestir… Se ha preferido esta versión a la última, *Makehuman 1.0 alpha 8*, porque la mayoría de los tutoriales de MakeClothes se refieren a ella.

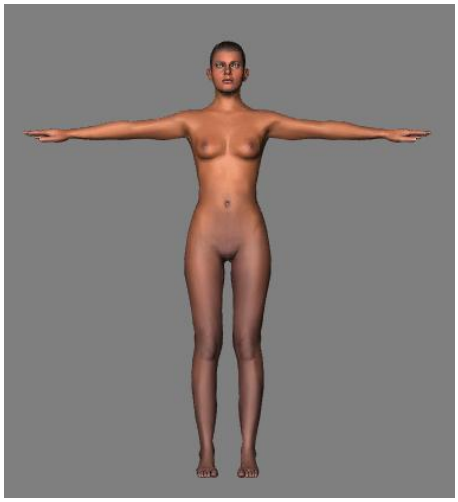


Figura 10: modelo inicial de Makehuman.

La parte más importante de este proyecto trata de conseguir los 19 datos (medidas, *measurements*) que necesitamos para realizar el maniquí con las medidas del usuario. Con estos datos, los introduciremos en Makehuman y se generara un maniquí. Estas medidas se encuentran en el apartado *Measure*, dentro de la pestaña *Modeling*.

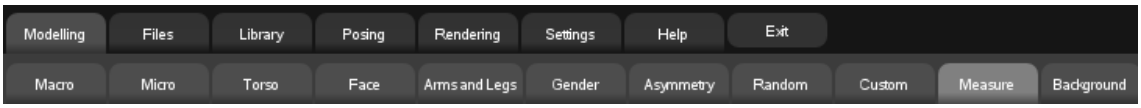


Figura 11: pestañas de Makehuman.

A continuación se muestran los datos métricos que se precisan para Makehuman. En los cuadros las medidas se pueden introducir de dos formas:

1. Haciendo click en la cifra que se quiere cambiar e introduciendo la cifra a mano,
2. Moviendo la barra hacia la izquierda para reducir o a la derecha para incrementar la cantidad.

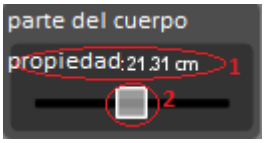


Figura 12: Cuadro ejemplo de introducción de medidas.

Estas son las medidas de costura que vamos a capturar de Kinect:

Cuello (neck):

- Anchura del cuello (neck circum).
- Altura del cuello (neck height).

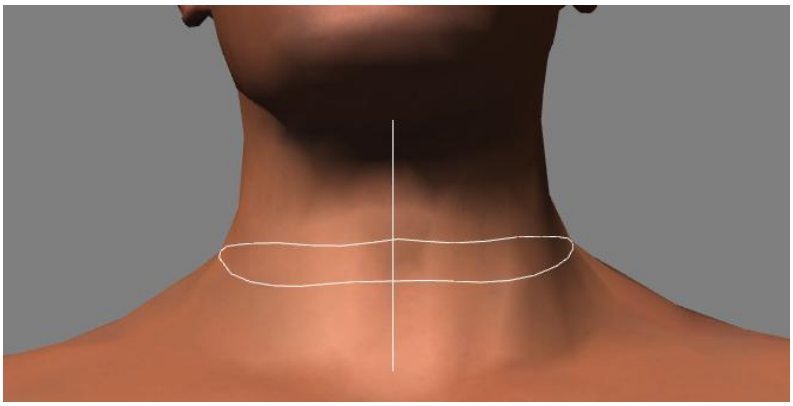


Figura 13: muestra de la anchura y altura del cuello.

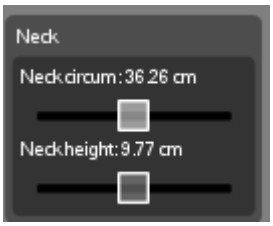


Figura 14: cuadro de cambio de medidas del cuello.

Parte superior del brazo (upper-arm):

- Anchura de la parte superior del brazo (upperarm circum).
- Longitud de la parte superior del brazo (upperarm lenght).

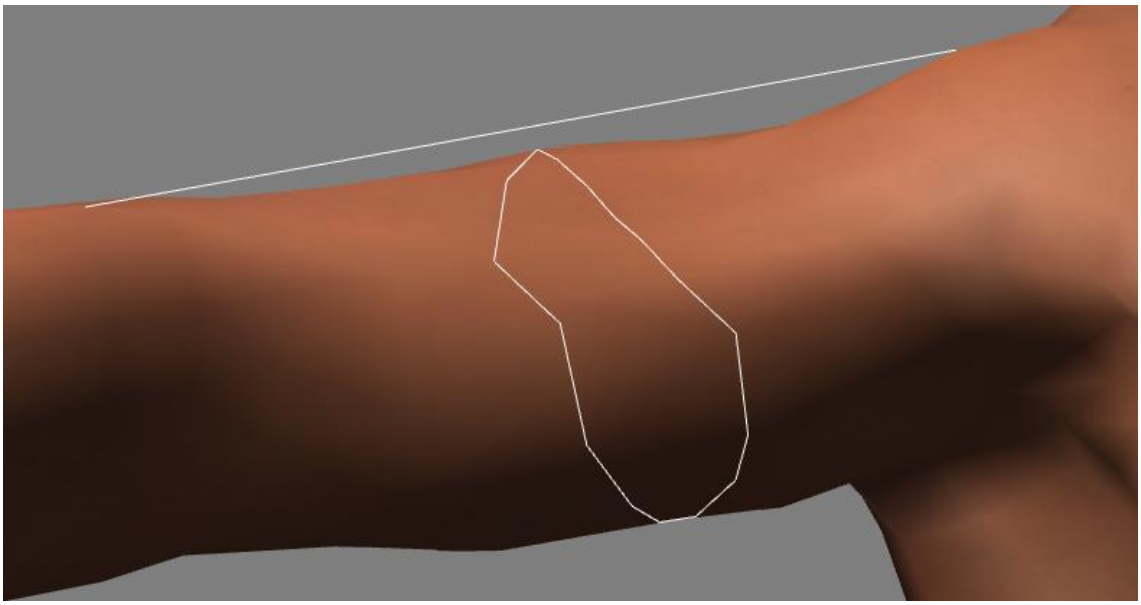


Figura 15: muestra de la anchura y altura del brazo superior.

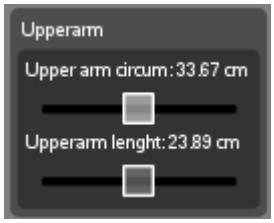


Figura 16: Cuadro de cambio de medidas del brazo superior.

Parte inferior del brazo (lower-arm)

- Longitud de la parte inferior del brazo (lowerarm lenght)
- Anchura de la muñeca (wrist circum)

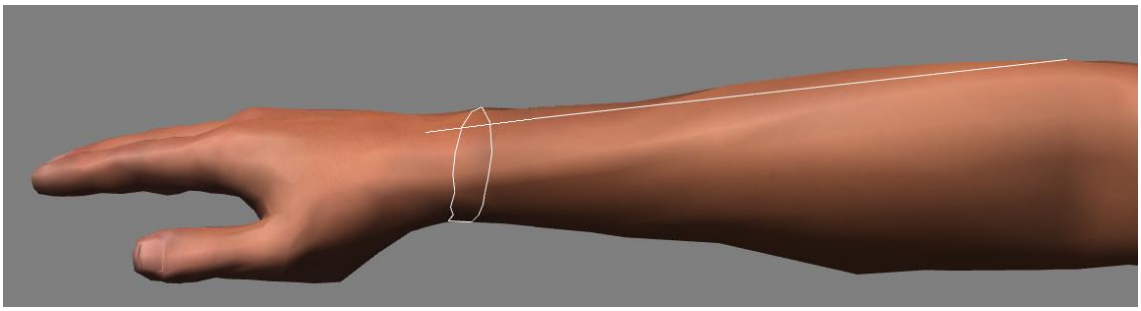


Figura 17: muestra de la anchura y altura del brazo inferior.

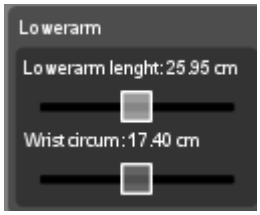


Figura 18: cuadro de cambio de medidas del brazo inferior.

Torso

1. Distancia de la parte frontal del pecho.
2. Anchura del pecho.
3. Anchura debajo del pecho.
4. Anchura de la cintura.
5. Distancia desde la nuca hasta la cintura.
6. Distancia desde la cintura hasta la cadera.
7. Distancia de la espalda.

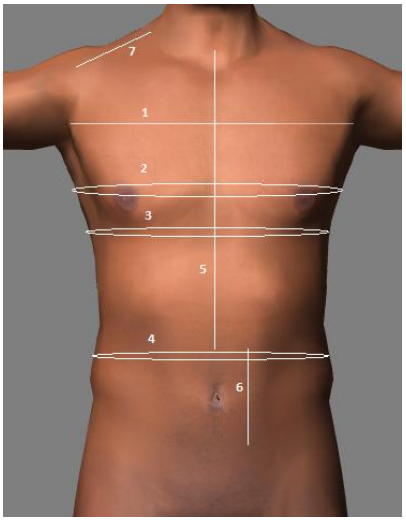


Figura 19: muestra de las medidas del torso.

Figura 20: cuadro de cambio de medidas del torso.

Anchura de la cadera

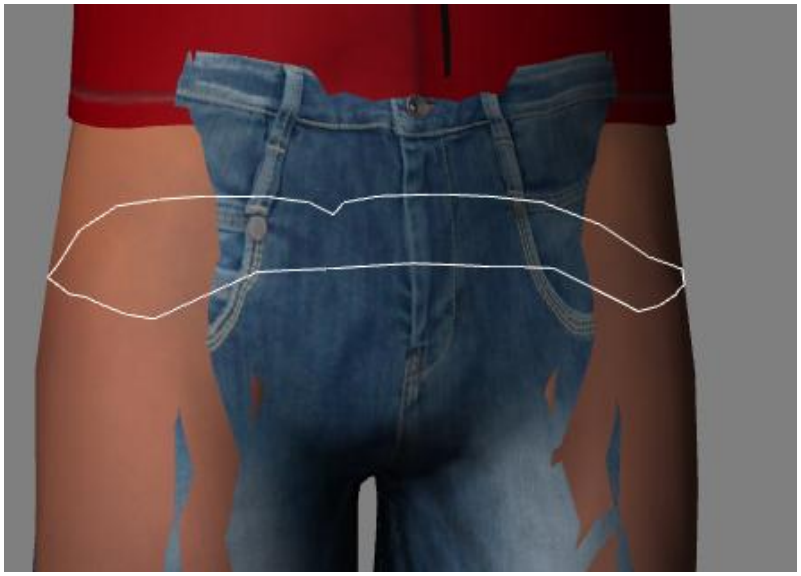


Figura 21: muestra de la anchura de la cadera.

Figura 22: cuadro de cambio de medida de la cadera.

Parte superior de la pierna

- Longitud de la parte superior de la pierna.
- Anchura del muslo.

Figura 23: muestra de la anchura y altura de la pierna superior.

Figura 24: cuadro de cambio de medidas de la pierna superior.

Parte inferior de la pierna:

- Longitud de la parte inferior de la pierna.
- Anchura del gemelo.

Figura 25: muestra de la anchura y altura de la pierna inferior.

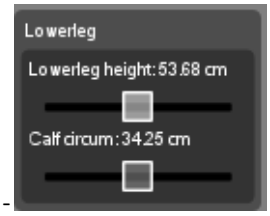


Figura 26: cuadro de cambio de medidas de la pierna superior.

Anchura del tobillo

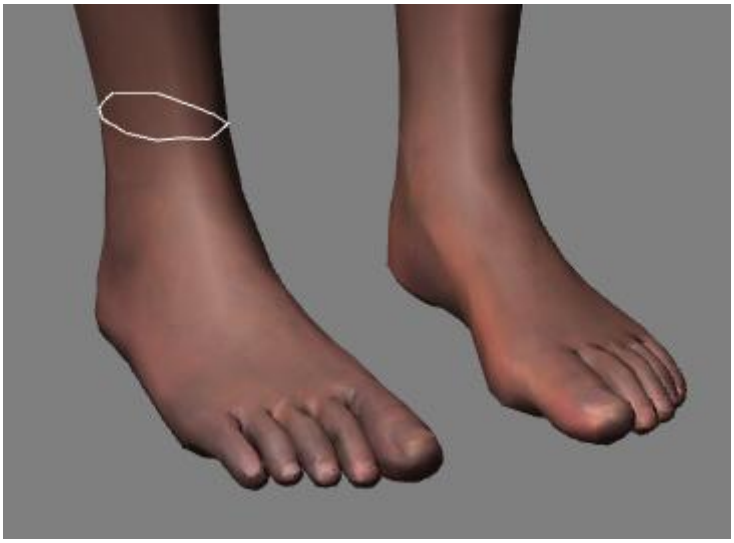


Figura 27: muestra de la anchura del tobillo.

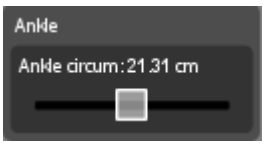


Figura 28: Cuadro de cambio de la medida del tobillo.

Obviamente estos datos los podemos conseguir manualmente midiendo con una cinta métrica. Pero se ha decidido realizar estas mediciones de manera automática. Para ello, utilizamos la Kinect.

10.2. Anexo B: Fase 2: Creación de prendas en Blender

10.2.1. Ropas en Makehuman

Esta es la ropa que existe en Makehuman. Se accede desde la pestaña ***clothes***, que está dentro de la pestaña ***library***. En esta colección de ropa, cualquier prenda puede colocarse al maniquí. Solamente hay que seleccionar la ropa y automáticamente queda ajustada al maniquí, sin importar su tamaño, si es que se elige esa opción. En otro caso, como ocurre en la figura 23, si la prenda es demasiado pequeña se aprecian en color carne las zonas que no llega a abarcar la prenda.

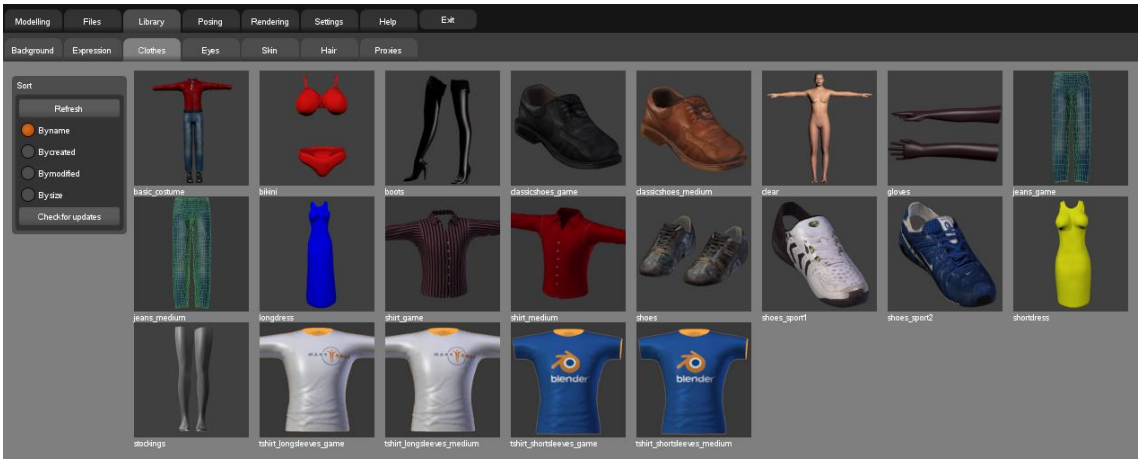


Figura 29: prendas existentes de Makehuman

Todas las prendas en Makehuman se componen de los siguientes componentes:

1. Archivo mhclo. Contiene información sobre la prenda: el nombre de los archivos que la componen, tipo de color (intensidad, reflejo de la luz, difusión). Al ser una prenda en tres dimensiones, se necesitan las coordenadas de todos los puntos que componen la prenda con los niveles de reflejo de la luz y de difusión del color.
2. Archivo obj: En este archivo se encuentran las coordenadas de los vértices que componen la prenda.
3. Imagen png: es la imagen del icono que se muestra en Makehuman: la imagen que el usuario debe pinchar.
4. Imagen png (máscara): la máscara es una imagen que oculta las partes del cuerpo que cubre la prenda.
5. Imagen de la textura. Es la textura la que se proyecta en el maniquí después de seleccionar la prenda.

Para empezar a crear ropa en tres dimensiones, se abre el programa Blender. Lo que se desea conseguir son los datos que proporcionan las prendas de Makehuman, crear ropa en tres dimensiones con la foto prenda. Esta foto estará compuesta por dos imágenes: La imagen delantera de la prenda y la imagen trasera.

Blender permite importar muchos tipos de archivos y entre ellos ha y uno que permite importar archivos ***.obj***. Por lo tanto lo que hacemos es importar el archivo obj de una prenda de -Makehuman. En este caso la prenda elegida será un pantalón.

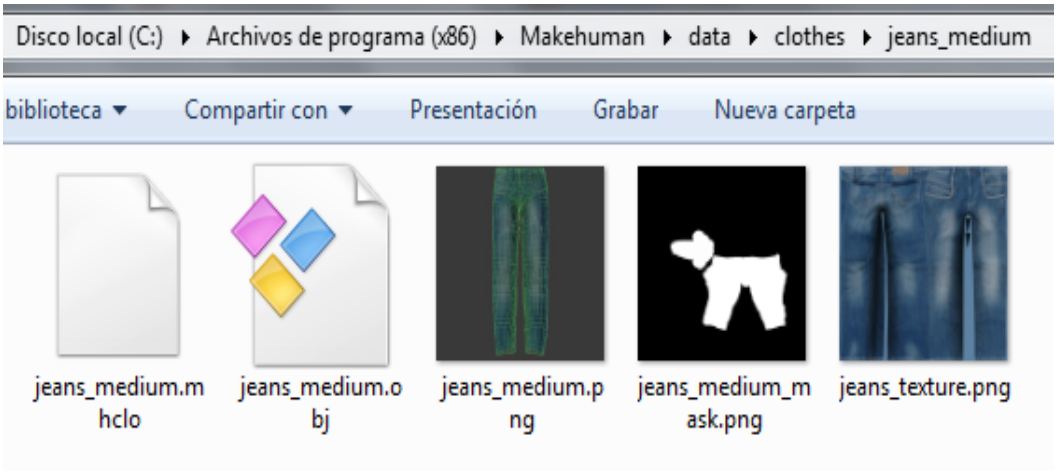


Figura 30: ejemplo de componentes de una prenda en Makehuman

10.2.2. Una forma de crear ropa en Blender.

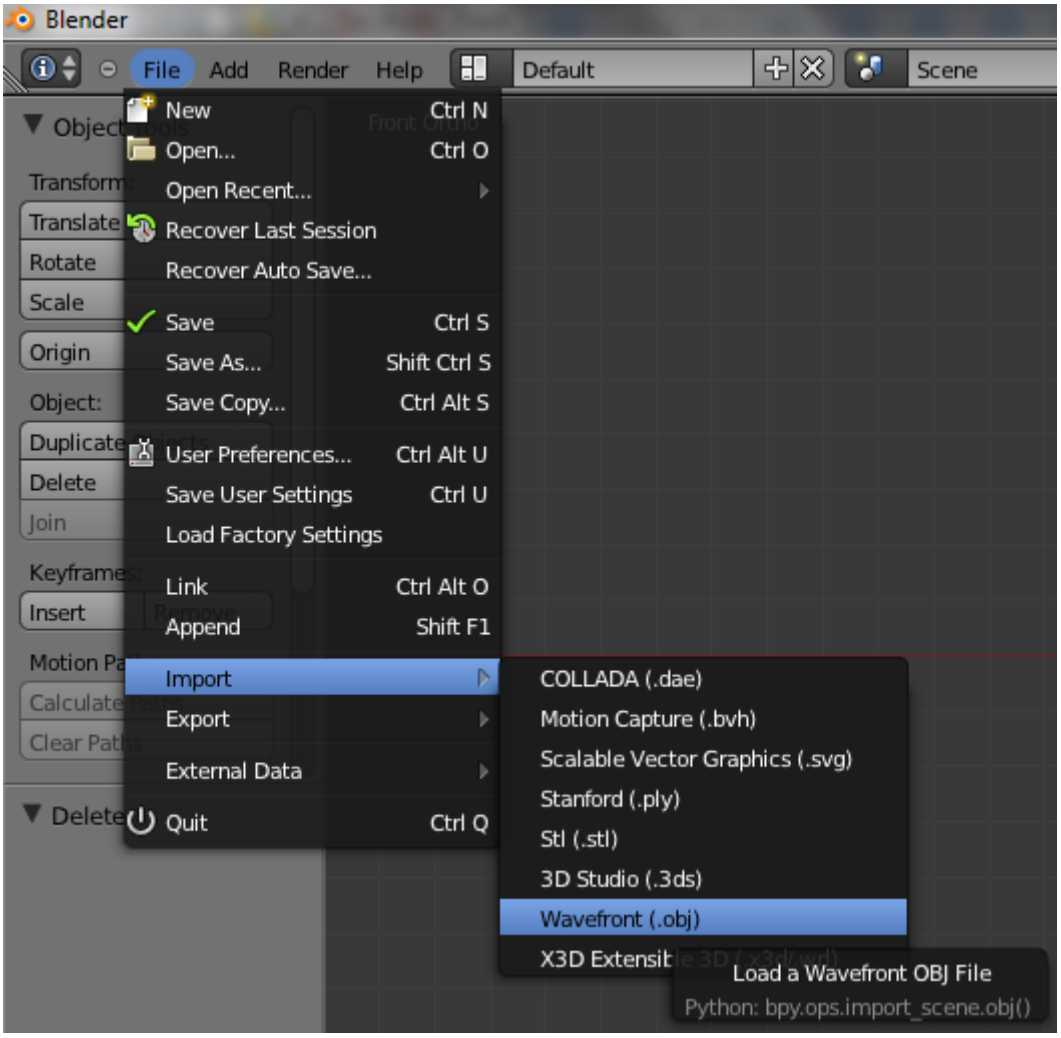


Figura 31: cargar un archivo obj en Blender.

El archivo que se importa es un pantalón en tres dimensiones que está representado por un conjunto de vértices conectados por aristas. Tres o cuatro vértices forman los límites de una cara. Una cara es una parte de malla que está rellena, y parecerá algo sólido cuando se *renderice*. Los vértices y aristas no se *renderizan* , sí las caras

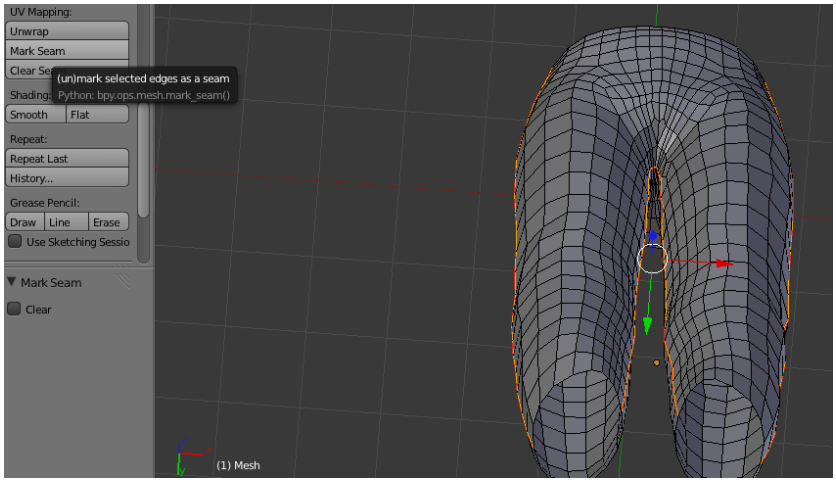


Figura 32: Muestra de un archivo de tipo obj en Blender

Una vez importada la malla o archivo obj de Makehuman vamos a pasar a implantar la foto real del pantalón en el objeto 3D. Existen muchas maneras de implantar una imagen, pero nos hemos decidido por el mapeado UV. Es una manera de mapear imágenes sobre objetos en tres dimensiones.

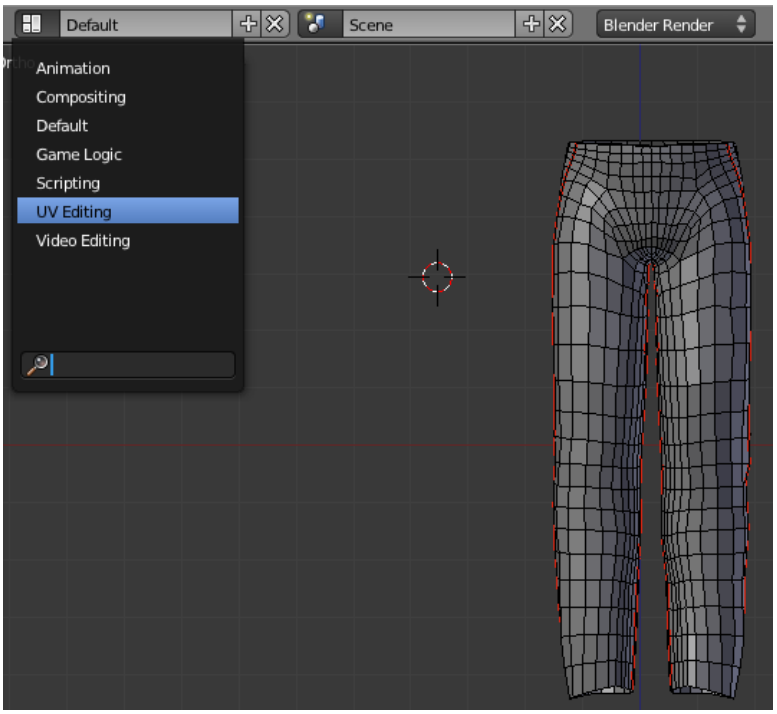


Figura 33: mapeado UV en Blender.

Este mapeado UV estira la malla para que el objeto 3D parezca una imagen. El objeto 3D está representado por un conjunto de coordenadas de tres dimensiones (XYZ), no obstante, el mapeado UV

está representado por coordenadas de dos dimensiones (UV). El proceso de crear el mapa UV se conoce como despliegue (en inglés ***unwrap***).

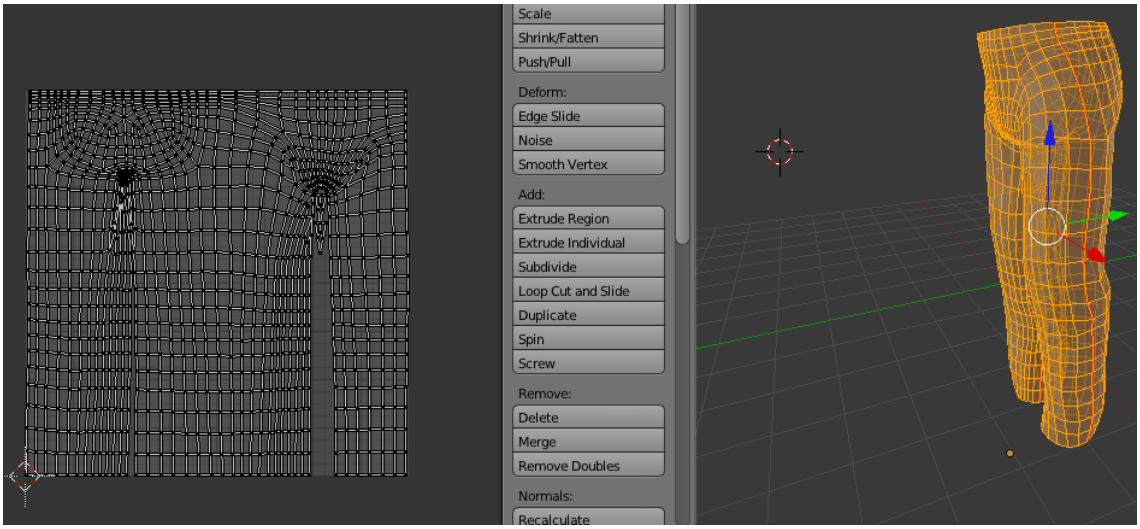


Figura 34: prenda en tres dimensiones y su mapa UV.

Como hemos dicho anteriormente, vamos a tener la fotografía de una prenda: la imagen frontal y la trasera. Como se puede apreciar en la imagen superior, vamos a tener que adecuar el mapa UV del objeto a la forma de la imagen.

Figura 35: fotografía de una prenda.

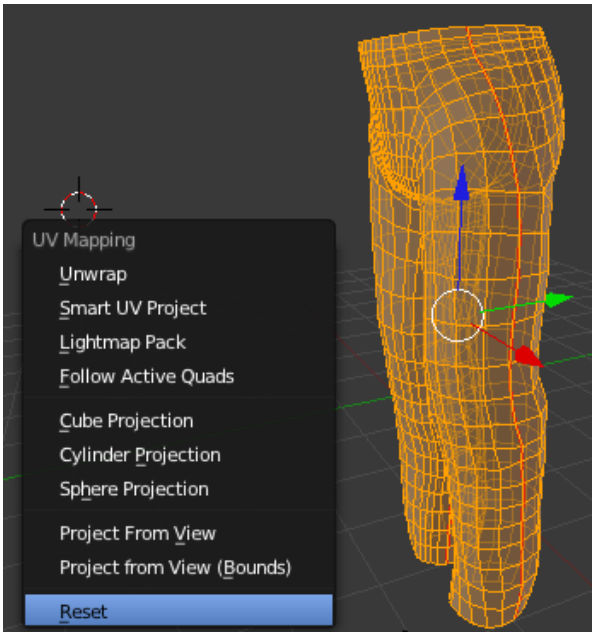


Figura 36: paso previo a división del mapa UV en dos piezas.

Con esta acción, el mapa del pantalón que antes era una sola pieza ahora va a estar separado en dos piezas, justo como las dos imágenes de la prenda que se presentan en la siguiente imagen.

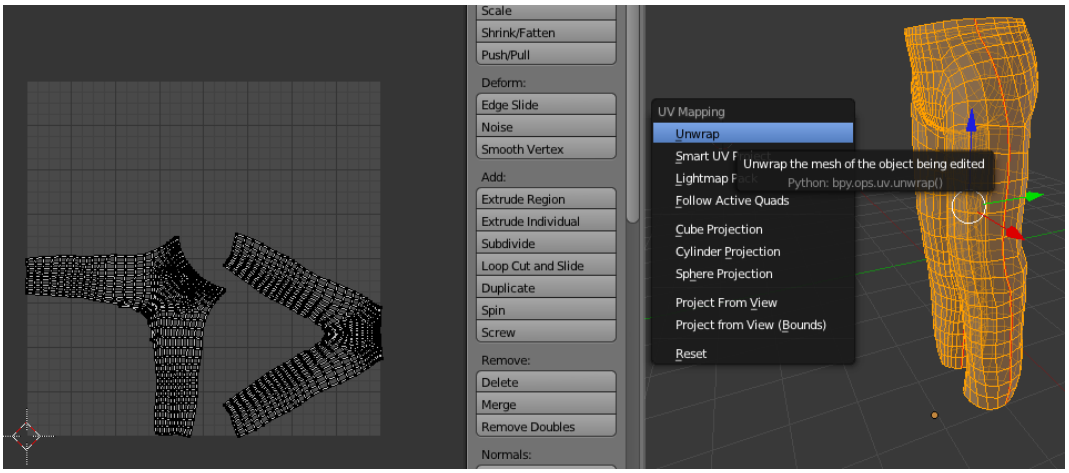


Figura 37: Prenda en tres dimensiones y su mapa UV dividido en dos piezas.

Ahora solo necesitamos abrir la imagen de la prenda para luego adecuarla al mapa o despliegue.

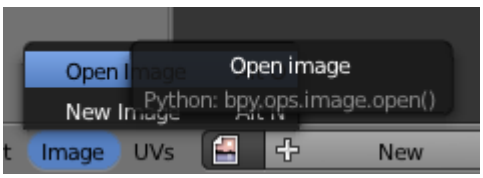


Figura 38: abrir imagen de la prenda en mapa UV.

Figura 39: mapa UV y fotografía de la prenda.

Una vez abierta la imagen, tendremos que adecuar el despliegue. Esto lo haremos utilizando las herramientas de transformación: Traslación, escalado y rotación.

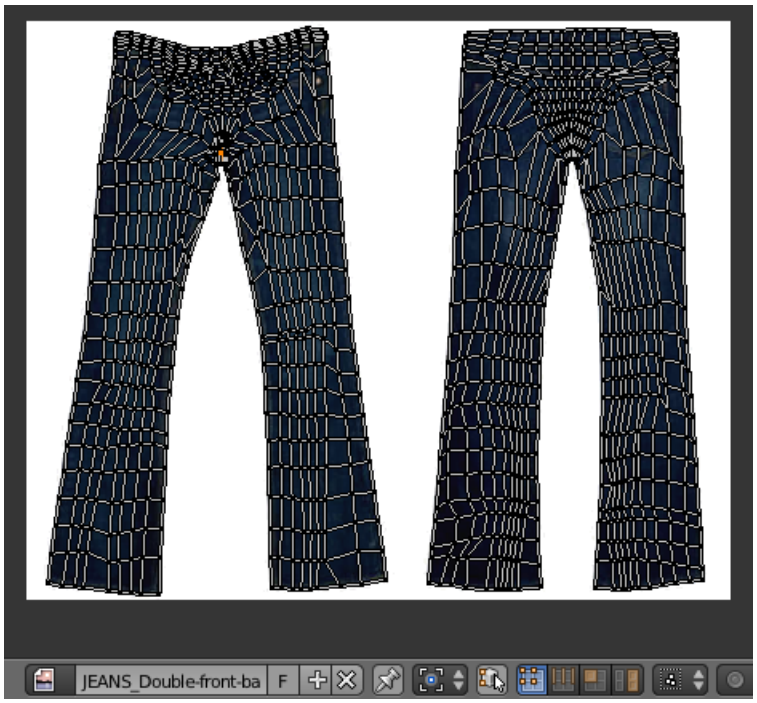


Figura 40: Mapa UV adecuado a la fotografía de la prenda.

Una vez adecuado el mapa a la imagen, obtenemos la siguiente imagen del objeto en 3D.

Figura 41: vista de la prenda en tres dimensiones con la fotografía en Blender.

Con este objeto en 3D tenemos el archivo obj que necesitamos para Makehuman. Para completar la prenda nos faltan los siguientes componentes:

1. La textura.
2. La imagen png que se debe mostrar al usuario para que escoja la prenda.
3. El archivo mhclo.

Para conseguir la imagen png (2) que se mostrará al usuario vamos a "renderizar" el objeto. Para ello aplicamos la textura de la imagen que hemos aplicado al mapa.

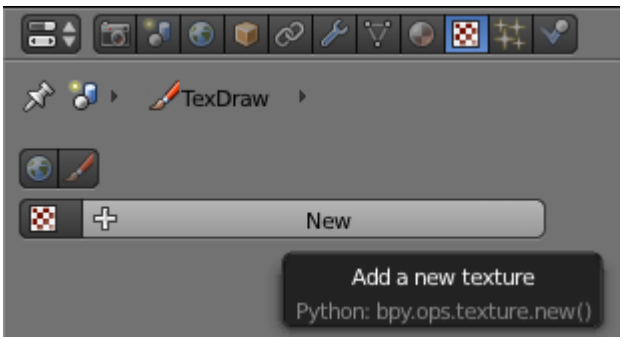


Figura 42: inserción de nueva textura.

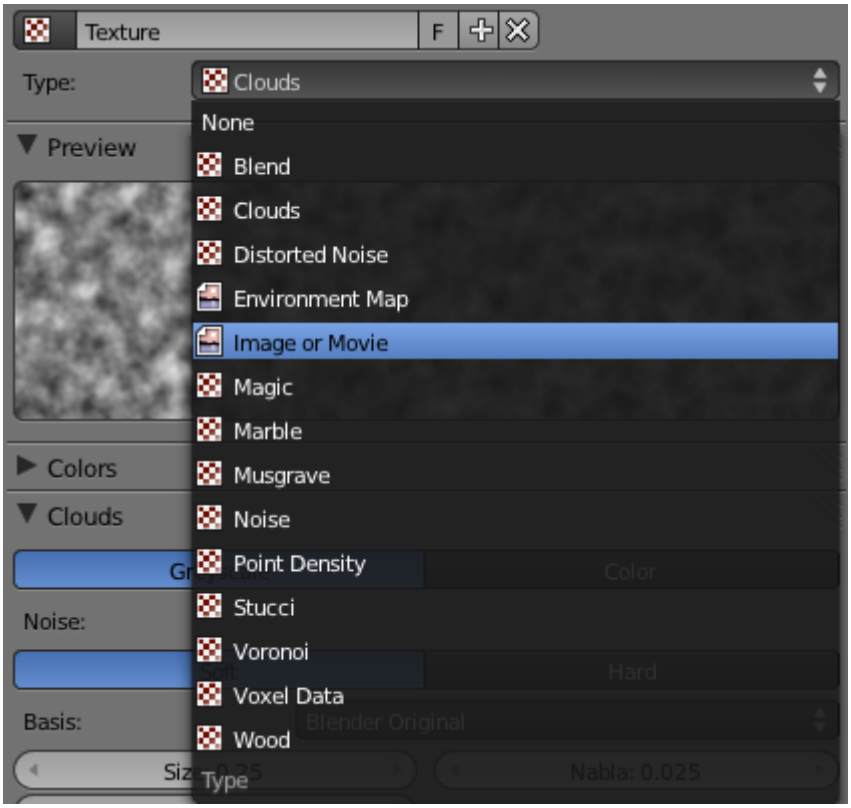


Figura 43: selección de imagen como textura del objeto 3D.

Elegimos que vamos a aplicarle una imagen a la textura.

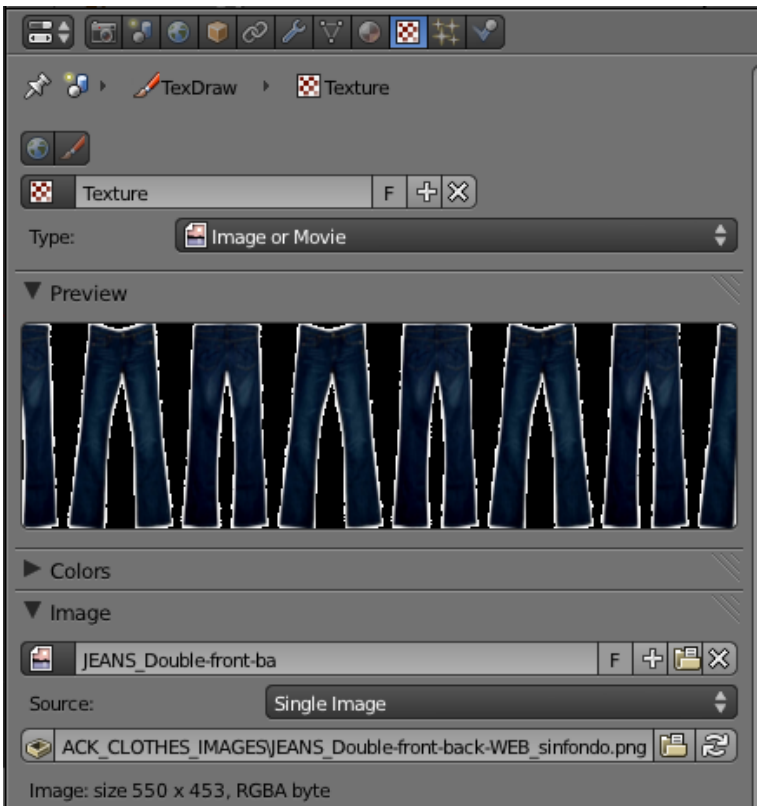


Figura 44: inserción de imagen como textura del objeto 3D.

Seleccionamos la imagen que vamos a aplicar.

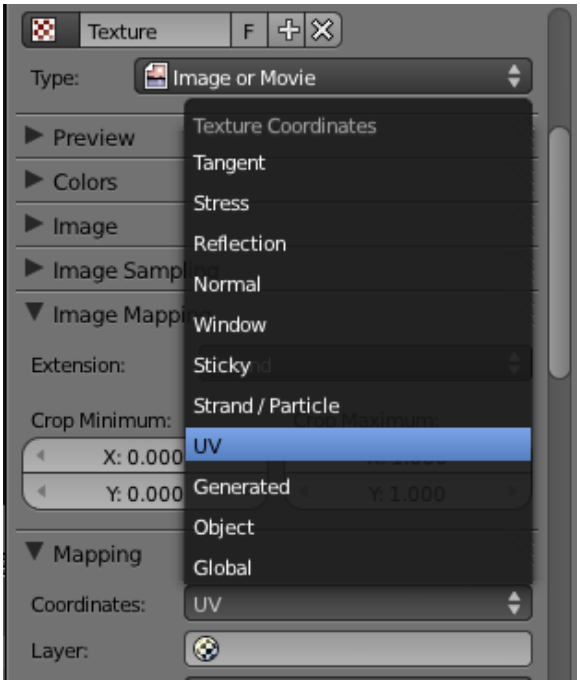


Figura 45: cambio de mapeado de la textura a mapeado UV.

Indicamos que la hemos conseguido del mapeado UV.

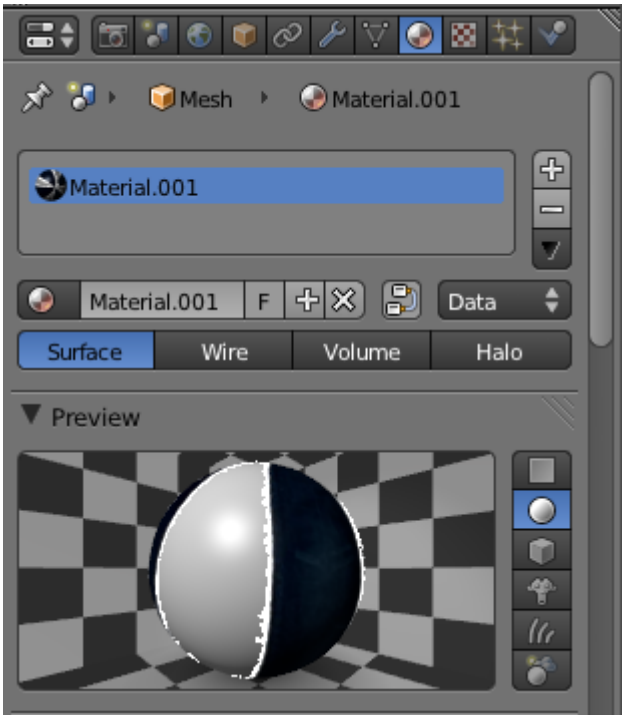


Figura 46: modificación del material del objeto 3D.

Indicamos que el material va a ser una textura y “renderizamos” la imagen. La tecla que hay que pulsar para “renderizar” es F10, la tecla F11 sirve para ocultar la escena del Render y F3 guarda la imagen creada por medio del Render.

Figura 47: Pantalón 3D renderizado. Resultado final de Blender.

Después de haber creado la prenda en 3D, se han presentado dos problemas:

- No hemos sabido crear la nueva mascara para Makehuman
- El archivo de tipo png que hemos creado para la textura que necesita Makehuman no era válido.

Hemos querido representar otra prenda en 3D siguiendo el mismo proceso para ver que es un proceso valido para crear ropa en 3D con imágenes reales. En este ejemplo vamos a crear una camiseta.

Figura 48: fotografía de la camiseta introducido en el mapa UV del objeto 3D.

Figura 49: mapa UV de la camiseta adecuado a la fotografía de la camiseta.

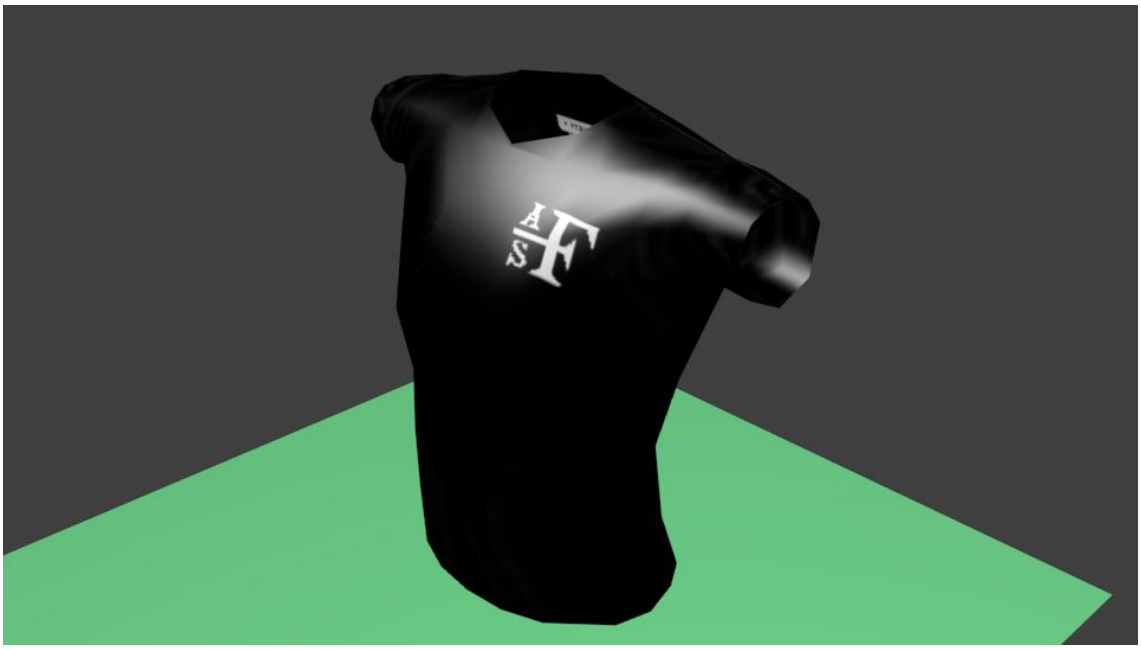


Figura 50: Camiseta 3D renderizada. Resultado final de Blender.

10.2.3. Otra forma de crear ropa utilizando Blender.

Otra forma de crear prendas en tres dimensiones, consiste en utilizar los add-ons. Los add-ons son herramientas que utiliza Blender pero no están instaladas por defecto. El usuario debe instalarlos.

Para ello utilizamos un add-ons llamado Makeclothes. Este add-ons conecta los programas Makehuman y Blender. Existen ocho versiones de Makehuman y muchas versiones de Blender. Según la

página oficial de Makehuman, la versión de Makeclothes incluida en Makehuman alpha 7 (la versión que se está usando en este proyecto) ha sido diseñada para trabajar con Blender 2.69. Por lo tanto en este proyecto también se utiliza la versión 2.69 de Blender.

Para instalar Makeclothes necesitamos copiar la carpeta en una localización de la carpeta de Blender para que este último pueda trabajar con él. La carpeta está localizada en Makehuman/tools/blender26x/makeclothes.

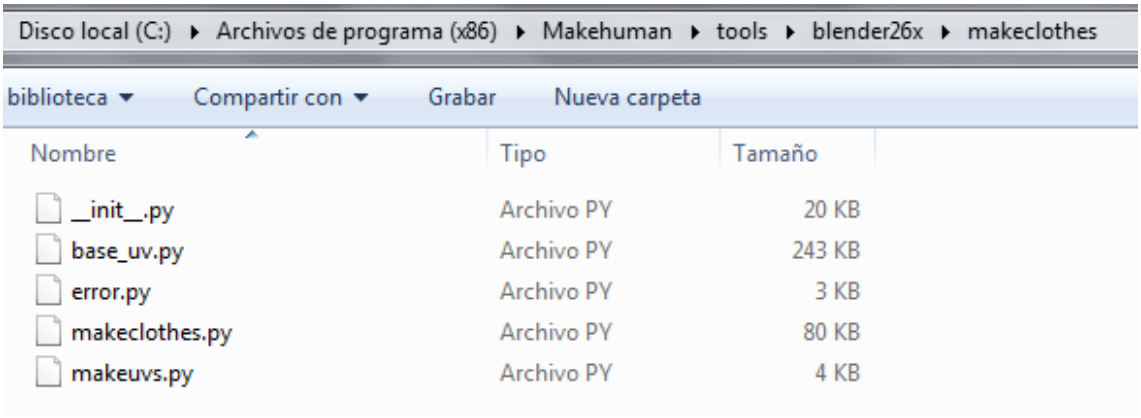


Figura 51: localización de archivos de Makeclothes.

Debemos copiar la carpeta entera en la siguiente carpeta:

C:\Usuarios\Admin\AppData\Roaming\Blender Foundation\Blender\2.6x

Figura 52: carpeta en la que debemos copiar Makeclothes.

Ahora podremos utilizar Blender con el add-on de Makeclothes instalado. Para ello debemos activar en Blender el add-on recién insertado. Iremos pues, a las preferencias de usuario, a la pestaña add-ons y seleccionaremos Makeclothes.

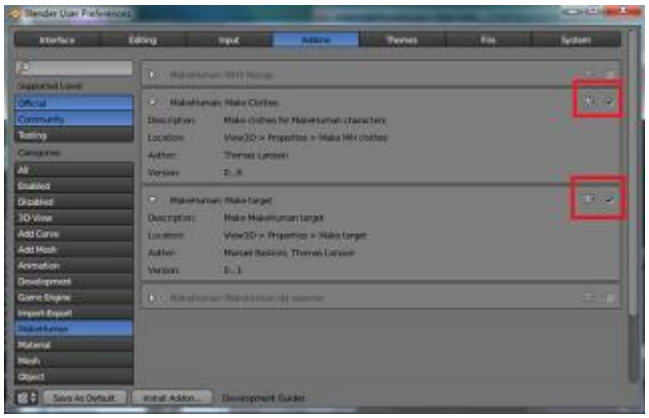


Figura 53: activación del *add-on Makeclothes*.

La pantalla de inicio de Blender se modifica, y aparece por defecto un maniquí vestido.

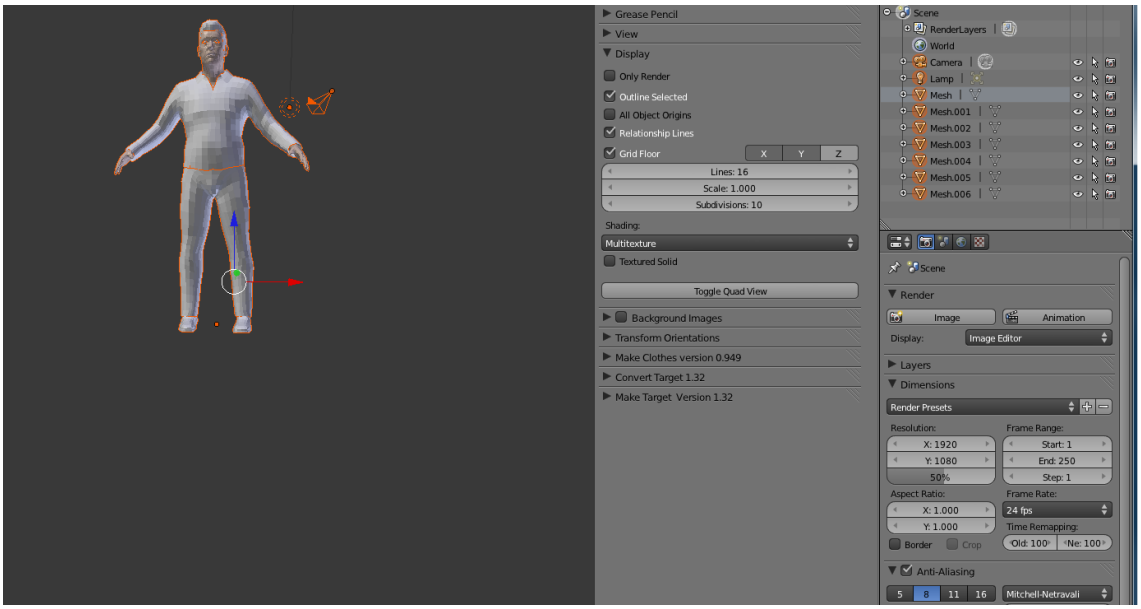


Figura 54: pantalla de inicio de Blender utilizando Makeclothes.

Es posible elegir el tipo de maniquí que queremos que se muestre. Un humano base sin vestir, un hombre caucásico, una mujer caucásica, un niño caucásico, un bebe caucásico…

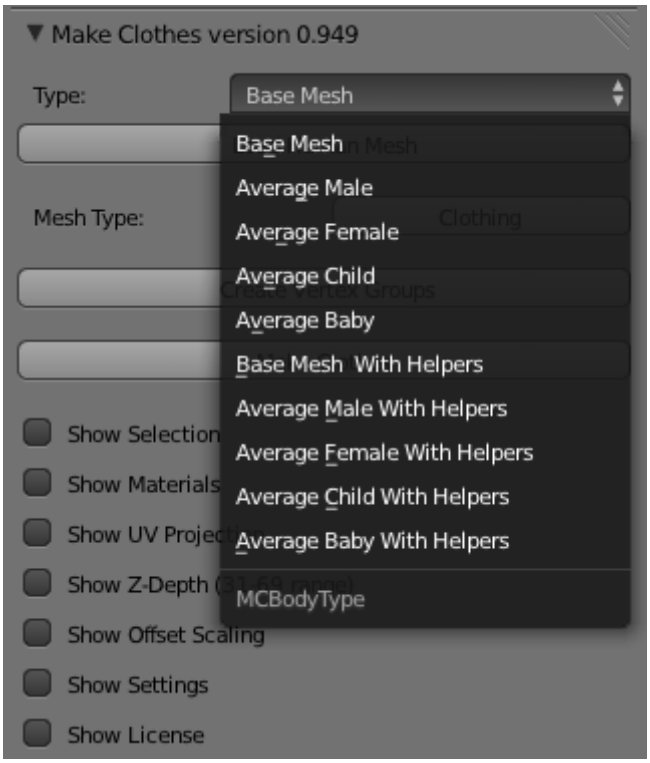


Figura 55: elección de maniquí a cargar en Blender.

En la imagen anterior, se puede comprobar en la parte superior derecha, que el maniquí está compuesto por diferentes mallas (*mesh*). Una vez hemos seleccionado la prenda que queremos modelar, le cambiamos el color. Y una vez modelado, subimos la prenda creada a Makehuman. ¡Vaya! nos da un error.

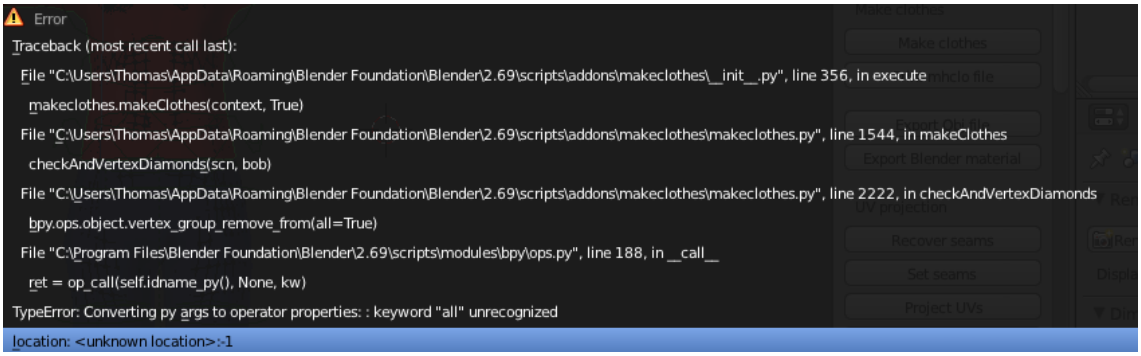


Figura 56: error en Makeclothes.

No hemos sido capaces de solucionarlo. Hemos probado con otras versiones de Blender pero hemos tenido la misma suerte.

Esta parte queda por desarrollar en el futuro.

10.3. ANEXO C: Fase 3: Calcular medidas a partir de Kinect

10.3.1. Calcular los datos para las medidas que necesita Makehuman

Una vez hemos obtenido los datos de Kinect, vamos a realizar cálculos sobre estos datos para conseguir las siguientes medidas que necesita Makehuman. En total necesitamos conseguir **19 medidas**. Para empezar, vamos a separar las medidas que necesitamos en tres categorías: longitudes (lenght), anchuras (width) y medidas del torso.

10.3.1.1. Calculo de longitudes

De la Kinect recibimos los siguientes datos: *jointCoordinates*. Es una matriz en la que se que recibe las coordenadas de las 20 articulaciones en metros desde el sensor hasta el píxel.

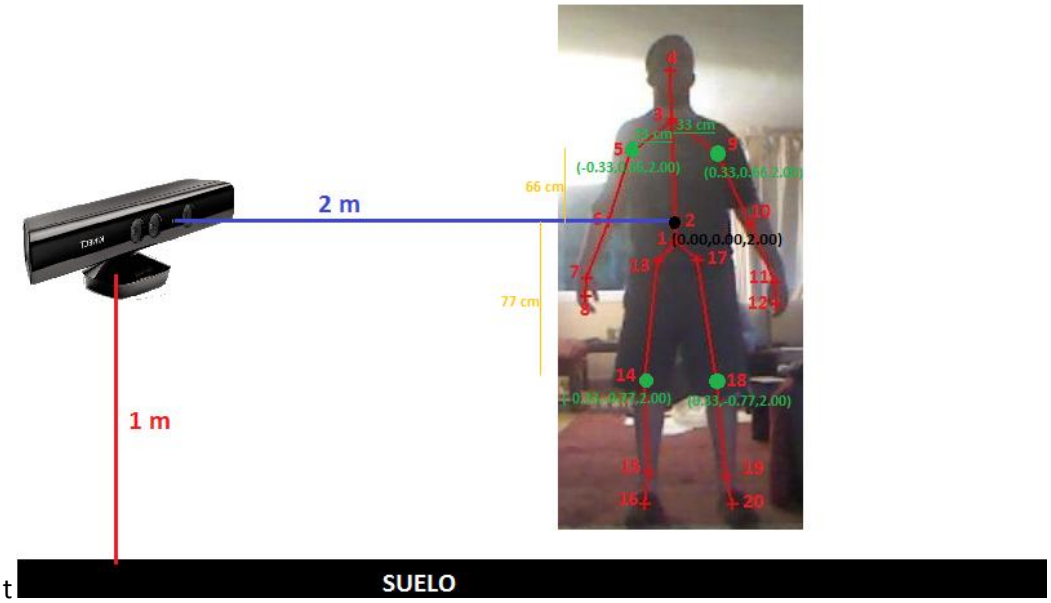


Figura 57: ejemplo de coordenadas reales (jointcoordinates) obtenidas por Kinect.

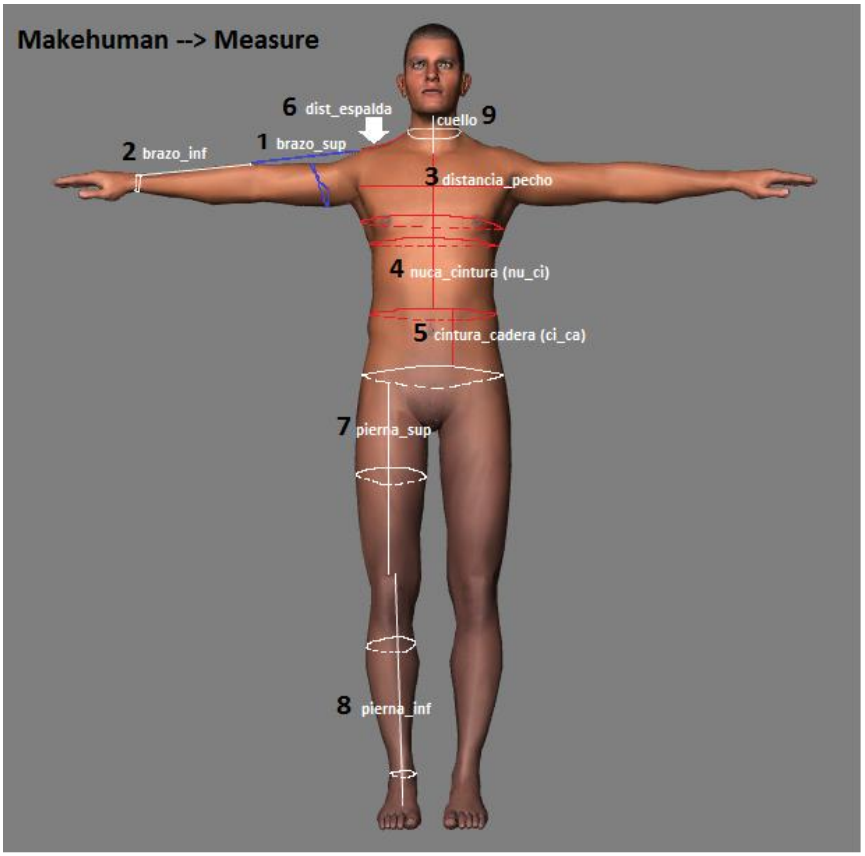


Figura 58: imagen de las 9 longitudes que necesitamos de Makehuman.

Figura 59: imagen de las 9 longitudes que obtendremos de Kinect.

El cuerpo no es uniforme. Las extremidades, por ejemplo, tienen diferentes longitudes. Por este motivo, calculamos las longitudes de las dos extremidades y luego computamos la media.

Para hallar la distancia entre dos puntos en cualquier espacio de dos coordenadas:

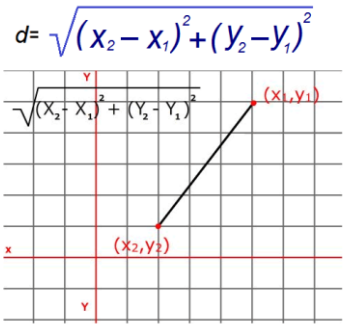


Figura 60: Ejemplo de distancia entre dos puntos. (*)

(*) Fuente: http://clasesmatematicas.blogspot.com.es/

Las longitudes que vamos a calcular son las siguientes:

1. Longitud brazo superior (*upper arm lenght*):

   Tenemos que calcular primero la distancia del brazo superior derecho: es la distancia entre los puntos de la coordenada de la articulación 5 (jointCoordinates(5)) y la articulación 6 (jointCoordinates(6)).

   Después tenemos que calcular la distancia del brazo superior izquierdo: es la distancia entre los puntos de la coordenada de la articulación 9 (jointCoordinates(9)) y la articulación 10 (jointCoordinates(10)).

   Para finalizar calculamos la media de estas dos distancias previamente calculadas. El resultado nos da una distancia en metros, así que lo multiplicamos por 100 para convertirlo a centímetros.

```
%%%% upperarm lenght
 upperarm_left =sqrt(((jointCoordinates(5,1)-(jointCoordinates(6,1)))^2)+(jointCoordinates(5,2)-(jointCoordinates(6,2)))^2);
 upperarm_rigth =sqrt(((jointCoordinates(9,1)-(jointCoordinates(10,1)))^2)+(jointCoordinates(9,2)-(jointCoordinates(10,2)))^2);

 upperarmlenght = (mean([upperarm_left,upperarm_rigth]))*100;
```

Código 1: calculo de longitud del brazo superior.

2. Longitud brazo inferior (*lower arm lenght*)

   Calculamos la distancia del brazo inferior derecho: es la distancia entre los puntos de la coordenada de la articulación 6 (jointCoordinates(6)) y la articulación 7 (jointCoordinates(7)).

   Después calculamos la distancia del brazo inferior izquierdo: es la distancia entre los puntos de la coordenada de la articulación 10 (jointCoordinates(10)) y la articulación 11 (jointCoordinates(11)).

Para finalizar calculamos la media de estas dos distancias previamente determinadas. El resultado nos da una distancia en metros, así que lo multiplicamos por 100 para convertirlo a centímetros.

```
%%%%% lowerarm lenght
 lowerarm_left =sqrt(((jointCoordinates(6,1)-(jointCoordinates(7,1)))^2)+(jointCoordinates(6,2)-(jointCoordinates(7,2)))^2);
 lowerarm_rigth = sqrt(((jointCoordinates(10,1)-(jointCoordinates(11,1)))^2)+(jointCoordinates(10,2)-(jointCoordinates(11,2)))^2);

 lowerarm = (mean([lowerarm_left,lowerarm_rigth]))*100;
```

Código 2: cálculo de longitud del brazo inferior.

3. Distancia parte frontal del pecho (*front chest dist*)

Esta línea no aparece en el esqueleto dibujado por Kinect, por lo que es una distancia que inferimos. Es la distancia entre los puntos de la coordenada de la articulación 5 (jointCoordinates(5)) y la articulación 9 (jointCoordinates(9)).

```
 %%%% front chest distance
frontchestdist = (sqrt(((jointCoordinates(5,1)-(jointCoordinates(9,1)))^2)+(jointCoordinates(5,2)-(jointCoordinates(9,2)))^2))*100;
```

Código 3: cálculo de longitud de la parte frontal del pecho.

4. Longitud nuca cintura (*nape to waist*)

Es la distancia entre los puntos de la coordenada de la articulación 2 (jointCoordinates(2)) y la articulación 3 (jointCoordinates(3)).

```
%%%% nape to waist
nape_to_waist = (sqrt(((jointCoordinates(3,1)-(jointCoordinates(2,1)))^2)+(jointCoordinates(3,2)-(jointCoordinates(2,2)))^2))*100;
```

Código 4: cálculo de longitud de la distancia desde la nuca a la cintura.

5. Longitud cintura cadera (*waist to hip*)

Es la distancia entre los puntos de la coordenada de la articulación 2 (jointCoordinates(2)) y la articulación 1 (jointCoordinates(1)).

```
%%%% waist to hip
waist_to_hip = (sqrt(((jointCoordinates(2,1)-(jointCoordinates(1,1)))^2)+(jointCoordinates(2,2)-(jointCoordinates(1,2)))^2))*100;
```

Código 5: cálculo de distancia desde la cintura a la cadera.

6. Distancia espalda (*shoulder dist)*

La distancia de la espalda es la distancia desde el final del cuello hasta el hombro.

Calculamos la distancia de la espalda derecha: es la distancia entre los puntos de la coordenada de la articulación 3 (jointCoordinates(3)) y la articulación 5 (jointCoordinates(5)).

Calculamos la distancia de la espalda izquierda: es la distancia entre los puntos de la coordenada de la articulación 3 (jointCoordinates(3)) y la articulación 9 (jointCoordinates(9)).

Para finalizar calculamos la media de estas dos distancias previamente determinadas. El resultado nos da una distancia en metros, así que lo multiplicaremos por 100 para convertirlo a centímetros.

```
%%%% shoulder dist
shoulder_dist_right = sqrt(((jointCoordinates(3,1)-(jointCoordinates(5,1)))^2)+(jointCoordinates(3,2)-(jointCoordinates(5,2)))^2);
shoulder_dist_left = sqrt(((jointCoordinates(3,1)-(jointCoordinates(9,1)))^2)+(jointCoordinates(3,2)-(jointCoordinates(9,2)))^2);

shoulder_dist = (mean([shoulder_dist_left,shoulder_dist_right]))*100;
```

Código 6: cálculo de distancia desde cuello hasta el hombro.

7. Pierna superior (*upper leg height*)

Calculamos la distancia de la pierna superior derecha: es la distancia entre los puntos de la coordenada de la articulación 13 (jointCoordinates(13)) y la articulación 14 (jointCoordinates(14)).

Calculamos la distancia de la pierna superior izquierda: es la distancia entre los puntos de la coordenada de la articulación 17 (jointCoordinates(17)) y la articulación 18 (jointCoordinates(18)).

Para finalizar calculamos la media de estas dos distancias previamente determinadas. El resultado nos da una distancia en metros, así que lo multiplicaremos por 100 para convertirlo a centímetros.

```
%%%% upperleg lenght
upperleg_left = sqrt(((jointCoordinates(13,1)-(jointCoordinates(14,1)))^2)+(jointCoordinates(13,2)-(jointCoordinates(14,2)))^2);
upperleg_rigth = sqrt(((jointCoordinates(17,1)-(jointCoordinates(18,1)))^2)+(jointCoordinates(17,2)-(jointCoordinates(18,2)))^2);

upperleg = (mean([upperleg_left,upperleg_rigth]))*100;
```

Código 7: cálculo de la longitud de la pierna superior.

8. Pierna inferior (*lower leg height*)

Calculamos la distancia de la pierna inferior derecha: es la distancia entre los puntos de la coordenada de la articulación 14 (jointCoordinates(14)) y la articulación 15 (jointCoordinates(15)).

Calculamos la distancia de la pierna inferior izquierda: es la distancia entre los puntos de la coordenada de la articulación 18 (jointCoordinates(18)) y la articulación 19 (jointCoordinates(19)).

Para finalizar calculamos la media de estas dos distancias previamente determinadas. El resultado nos da una distancia en metros, así que lo multiplicaremos por 100 para convertirlo a centímetros.

```
%%%% lowerleg lenght
lowerleg_left = sqrt(((jointCoordinates(14,1)-(jointCoordinates(15,1)))^2)+(jointCoordinates(14,2)-(jointCoordinates(15,2)))^2);
lowerleg_rigth = sqrt(((jointCoordinates(18,1)-(jointCoordinates(19,1)))^2)+(jointCoordinates(18,2)-(jointCoordinates(19,2)))^2);

lowerleg = (mean([lowerleg_left,lowerleg_rigth]))*100;
```

Código 8: cálculo de la longitud de la pierna inferior.

9. Altura del cuello (*neck height*)

Es la distancia entre los puntos de la coordenada de la articulación 3 (jointCoordinates(3)) y la articulación 4 (jointCoordinates(4)). Esta distancia es el doble de la que necesitamos, por lo que la dividimos entre dos.

```
%%%% neck height
 neck_height = (sqrt(((jointCoordinates(3,1)-(jointCoordinates(4,1)))^2)+(jointCoordinates(3,2)-(jointCoordinates(4,2)))^2))*100/2;
```

Código 9: cálculo de la altura del cuello.

10.3.1.2. Calculo de anchuras

En esta sección, vamos a utilizar *JointImageIndices* que son los índices de las 20 articulaciones del usuario respecto a la imagen recogida por la cámara de color. Antes hemos comentado que en este proyecto se está utilizando la versión R2013a de MatLab y esto tiene una desventaja: no podemos utilizar *jointDepthIndices (*los índices de las 20 articulaciones del usuario respecto a la imagen recogida por la cámara de profundidad*)* que proporciona Kinect. Si pudiésemos utilizar este dato podríamos calcular los valores que necesitamos más fácilmente. Tendríamos que calcular dónde hay una diferencia entre los píxeles en la imagen de profundidad. La imagen de profundidad que devuelve la cámara, conocida como mapa de profundidad, tiene la misma cantidad de píxeles que una imagen en RGB, pero el valor del píxel es la distancia que hay entre el píxel y la cámara, es decir, la profundidad.

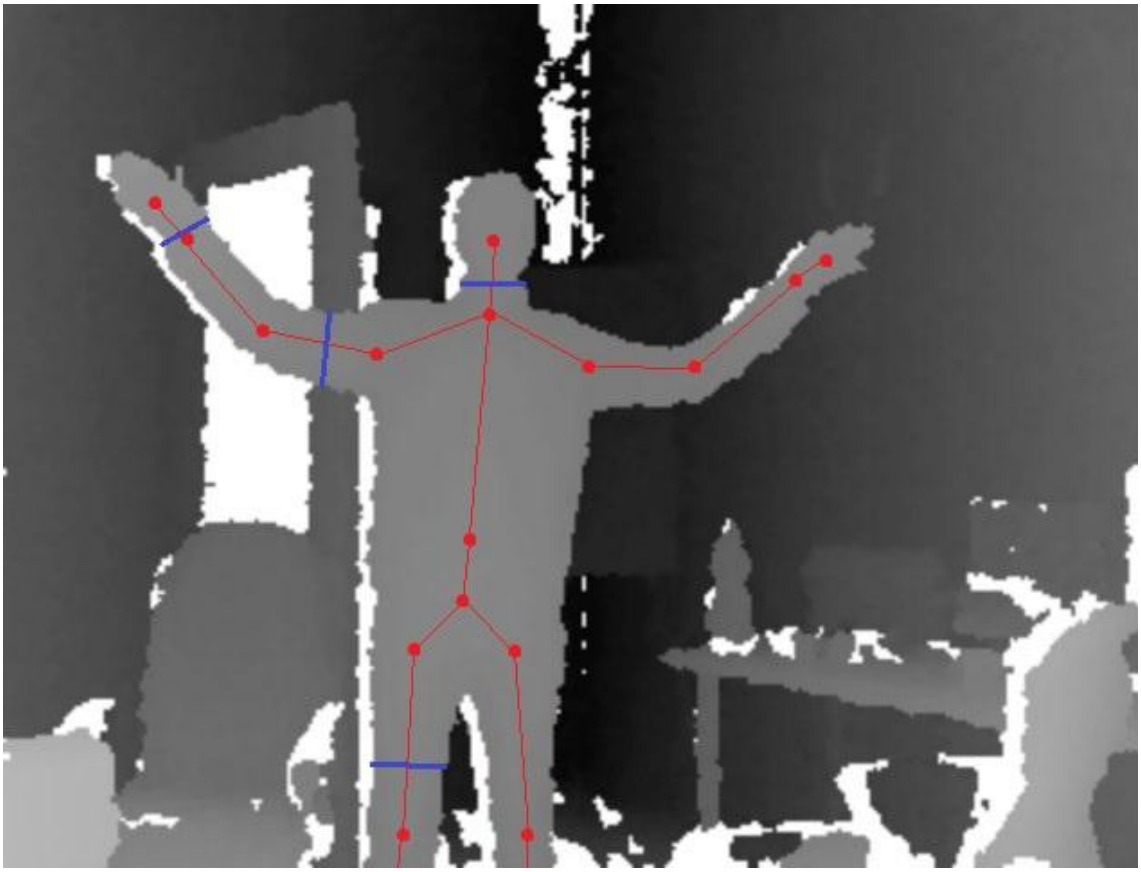


Figura 61: cálculo de las anchuras en la imagen de profundidad. (*)

(*) Fuente: http://www.mindtreatstudios.com/

He aquí un ejemplo con los datos que se podrían extrapolar. Supongamos que tenemos una pelota de 25 centímetros de diámetro a dos metros de distancia de la Kinect. Queremos calcular su perímetro. La pelota y el dispositivo están a la misma altura, por lo que prescindiremos de la coordenada Y.

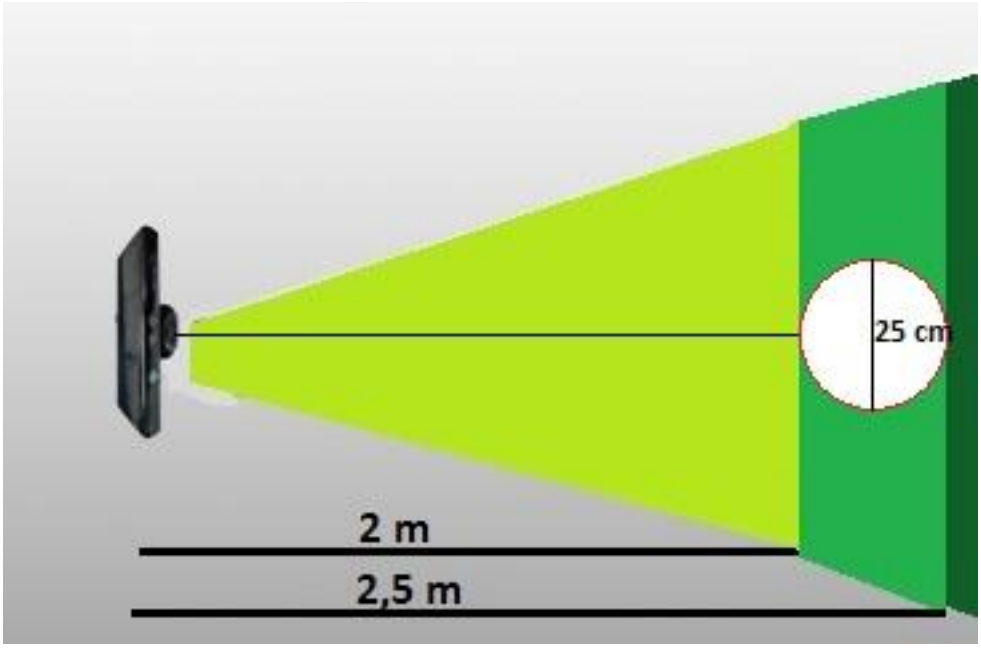


Figura 62: Plano picado de Kinect y una pelota. (*)

(*) Fuente: http://www.reallusion.com/

Estos serian los datos que recogería la cámara de profundidad.

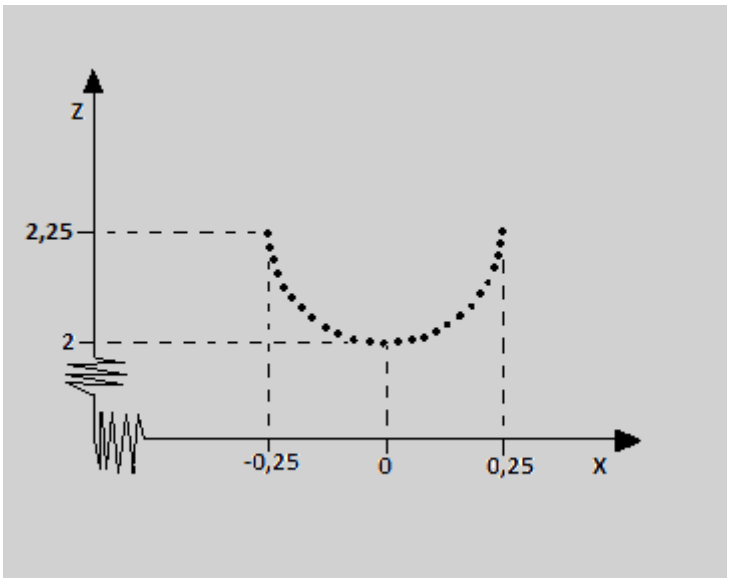


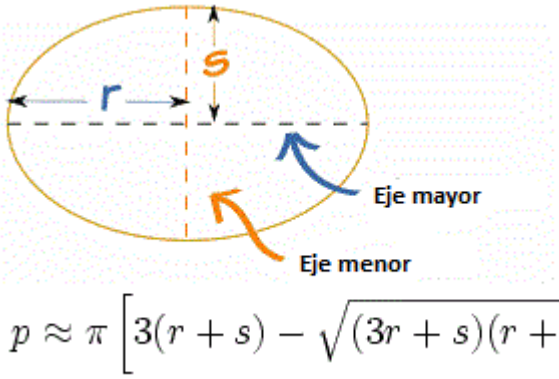
Figura 63: gráfico de la visión de profundidad de Kinect.

Nos interesan 3 puntos especialmente: (0,2), (-0.25, 2.5) y (-0.25, 2.5), es decir, los extremos que nos indicaran la magnitud de la pelota. Por lo tanto, inferimos que el radio de la pelota es de 25 cm y la pelota tendría un perímetro o circunferencia de 25*2*π = 157,07 centímetros.

Con la anchura de las extremidades que deseamos calcular ocurre algo similar. Quisiera añadir que las anchuras de las extremidades no son iguales. La forma del brazo superior o de la muñeca es elipsoide mientras que la forma del muslo o del gemelo es más bien circular. Esto significa que las fórmulas que tendríamos que aplicar para calcular su anchura o perímetro serían diferentes:

- Fórmula del cálculo del perímetro de una circunferencia: 2*π*Radio

- Fórmula del cálculo del perímetro de una elipse. Esta es una fórmula planteada por el matemático indio Ramunajan.



$$p \approx \pi \left[3(r+s) - \sqrt{(3r+s)(r+3s)} \right]$$

Figura 64: fórmula del perímetro la elipse según *Ramunajan*. (*)

(*) Fuente: http://www.disfrutalasmatematicas.com/

Descalibracion de Kinect

Como se puede apreciar en la siguiente imagen, hemos aplicado los índices de la imagen en RGB, es decir, *JointImageIndices* en el mapa de profundidad y se puede apreciar la *descalibración* de Kinect.

```
>
> util_skeletonViewer(jointIndices, prof, nSkeleton);
>
```

Código 10: Índices de la imagen RGB superpuestos a la imagen de profundidad.

Figura 65: muestra de descalibracion de Kinect.

El problema es el siguiente: Como inferimos de la imagen en color las anchuras que necesitamos. Tenemos que encontrar los siguientes valores:

1. Anchura de brazo superior.
2. Anchura de muñeca.
3. Anchura de muslo.
4. Anchura de gemelo.
5. Anchura de tobillo.
6. Anchura de cuello.

El resto de medidas que necesitamos, anchura de pecho, anchura de bajo pecho, anchura de cintura y anchura de cadera se encuentran en la siguiente sección.

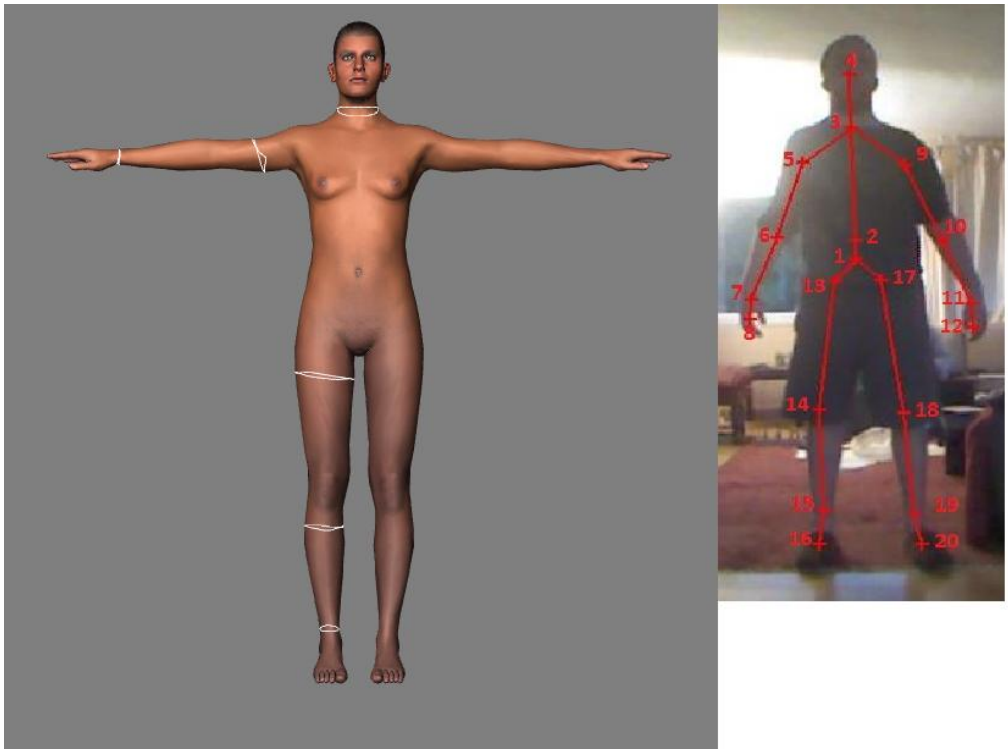


Figura 66: anchuras que necesitamos calcular e imagen RGB con los índices numerados.

Para calcular estas 6 anchuras vamos a procesar la imagen que nos proporciona la cámara de color y con ese tratamiento calculamos las medidas.

El tratamiento consiste en hallar la silueta o borde del usuario. En este proyecto, la cámara de color se centra únicamente en el usuario. *JointImageIndices* proporciona las coordenadas (x,y) de las articulaciones del usuario respecto a la imagen RGB. Queremos encontrar los píxeles que limiten la imagen del usuario. El formato de representación de color ofrecido por las imágenes RGB no es apropiado para encontrar el borde. Cada píxel tiene tres valores, Red, Green, Blue. Cada valor utiliza 8 bits, y esto supone que los valores están en un rango de números enteros entre 0 y 255. Esto significa que existen 256^3 = 1677216 colores distintos.

Por lo que *binarizaremos* la imagen para que los píxeles de la imagen solo tenga dos valores. MatLab, también es útil para el pre-procesamiento y procesamiento de imágenes. Esto se debe a *Image Processing Toolbox*. Vamos a pre-procesar la imagen en RGB para conseguir detectar el borde del usuario (silueta) con los siguientes pasos:

1. Pasar nuestra imagen con valores RGB a **escala de grises (*rgb2gray(imagen_RGB)*)**. Lo hace mediante la eliminación del tono y saturación, manteniendo la iluminación.
2. A continuación **ecualizamos la imagen (*histeq(imagen_grises,nivel)*)**. Transformamos la imagen en la escala de grises que tenemos. ***nivel*** es el número de niveles de intensidad de grises. En nuestro caso, la imagen de salida tendrá 100 valores distintos de grises. Para ver el resultado de la ecualización vamos a comparar los histogramas de las dos imágenes. Un histograma es la representación gráfica de la frecuencia relativa de los niveles de grises. Si el nuevo número de grises es menor que numero original de niveles de grises, *histeq* distribuirá los niveles para conseguir un histograma plano.

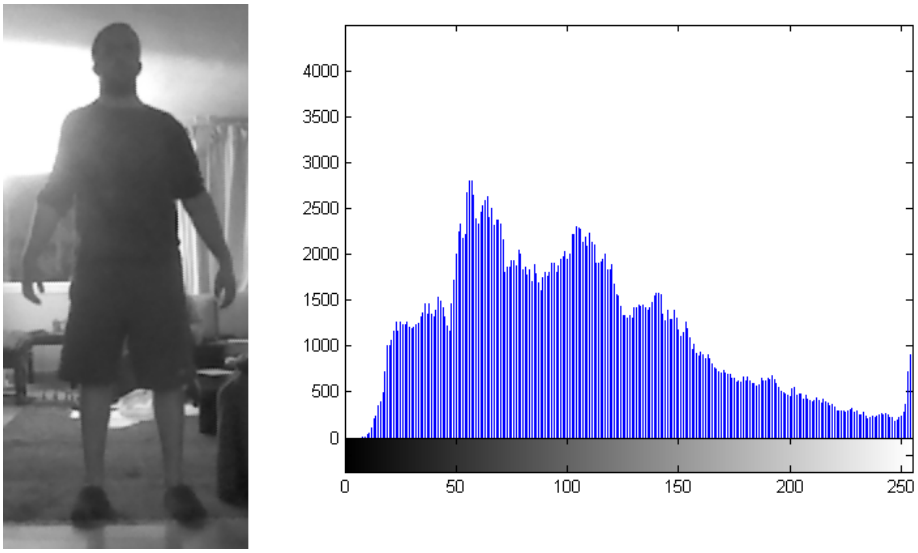


Figura 67: imagen en escala de grises y su histograma correspondiente.

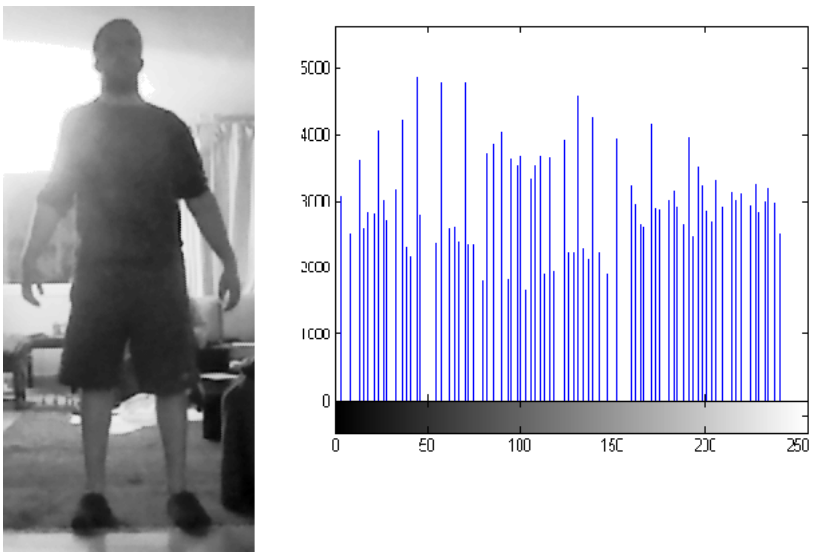


Figura 68: imagen en escala de grises ecualizada y su histograma correspondiente.

3. Utilizando el comando ***edge*** (borde) y el método ***sobel*** calculamos el valor del umbral de sensibilidad (***threshold***).
4. Creamos un factor que ajuste el valor del umbral.
5. Para finalizar, ajustamos el valor del umbral y utilizamos otra vez el comando ***edge*** para obtener una máscara binaria con un umbral más ajustado.

```
image_gray=rgb2gray(image);
g=histeq(image_gray,100);
[~, threshold] = edge(g, 'sobel');
fudgeFactor = .5;
BWs = edge(g,'sobel', threshold * fudgeFactor);

figure,imshow(BWs);
util_skeletonViewer(jointIndices, BWs, nSkeleton);
```

Código 11: cálculo de bordes e imagen de bordes con índices de articulaciones.

Como se puede observar en el histograma de la siguiente imagen, la nueva imagen solo tendrá dos valores: 0 y 1. 1 es el valor de los píxeles de los bordes y 0 es el valor para el resto de los píxeles.

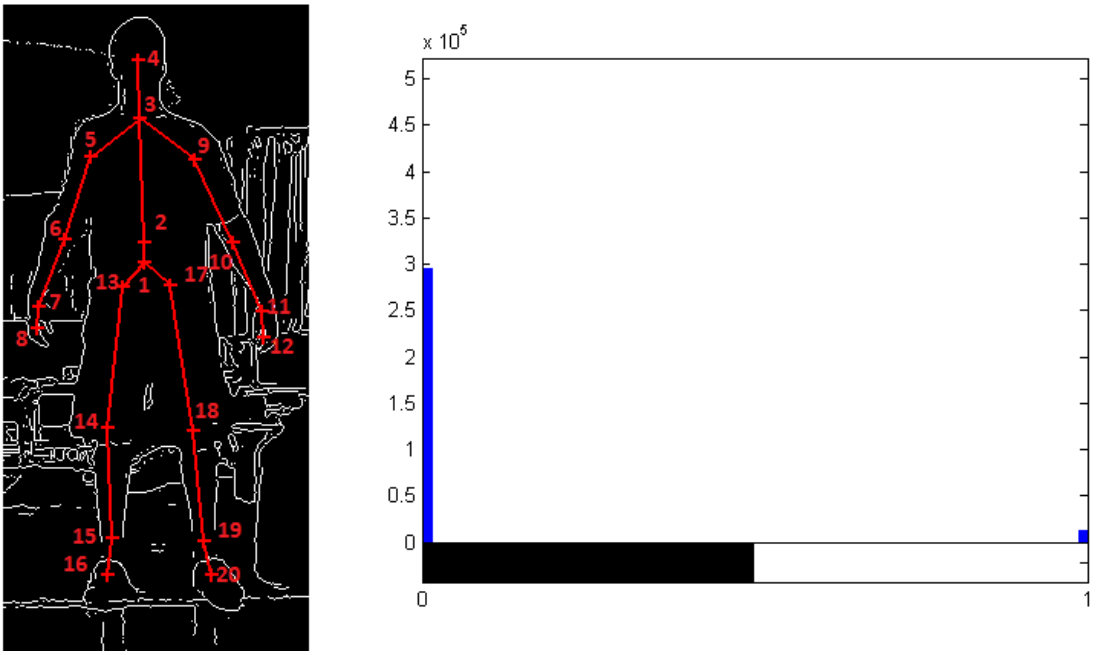


Figura 69: imagen de bordes con articulaciones y su histograma correspondiente.

Para que la que la detección de bordes sea satisfactoria, el usuario debe situarse en un fondo de un solo color y debemos usar prendas que tengan un color diferente al del fondo. Si no, la aplicación no será capaz de detectar los bordes ya que diferencia las tonalidades de los colores.

Una vez hemos detectado los bordes del usuario en la imagen que nos devuelve la cámara de color y tenemos las coordenadas de las 20 articulaciones en esa imagen, vamos a crear 4 funciones.

Todas ellas hacen lo mismo, medir la distancia de la línea de color rojo que crean los índices hasta el borde del usuario. Usamos el mismo método que hemos utilizado para las longitudes: como el cuerpo humano no es simétrico, medimos las anchuras de la parte izquierda y la parte derecha del cuerpo y después calcularemos la media.

He aquí la tabla que relaciona los índices, las anchuras que necesitamos y los nombres de las funciones que hemos creado.

Nombre anchura	Nombre función	Índices de JointImageIndices
Brazo superior derecho (Upper	Anchura1	5, 6

arm right)		
Brazo superior izquierdo (Upper arm left)	Anchura2	9, 10
Muslo derecho (thigh right)	Anchura1	13, 14
Muslo izquierdo (thigh left)	Anchura2	17, 18
Gemelo derecho (calf right)	Anchura1	14, 15
Gemelo izquierdo (calf left)	Anchura2	18, 19
Cuello derecho (neck circ right)	Anchura1	3,4
Cuello izquierdo (neck circ left)	Anchura2	3,4
Muñeca derecha (wrist right)	Anchura3	6, 7
Muñeca izquierda (wrist left)	Anchura4	10, 11
Tobillo derecho (ankle right)	Anchura3	14, 15
Tobillo izquierdo (ankle left)	Anchura4	18, 19

Tabla 4: tabla que relaciona los índices de imagen RGB, las anchuras y los nombres de las funciones.

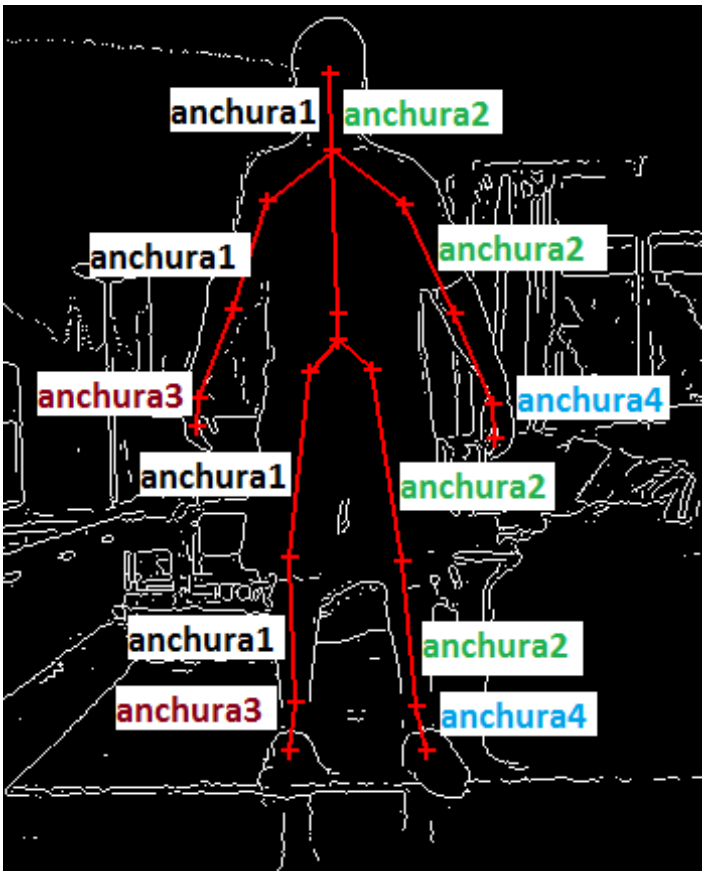


Figura 70: localización exacta donde se aplican las funciones.

Cuando todas las funciones se aplican obtenemos la siguiente imagen:

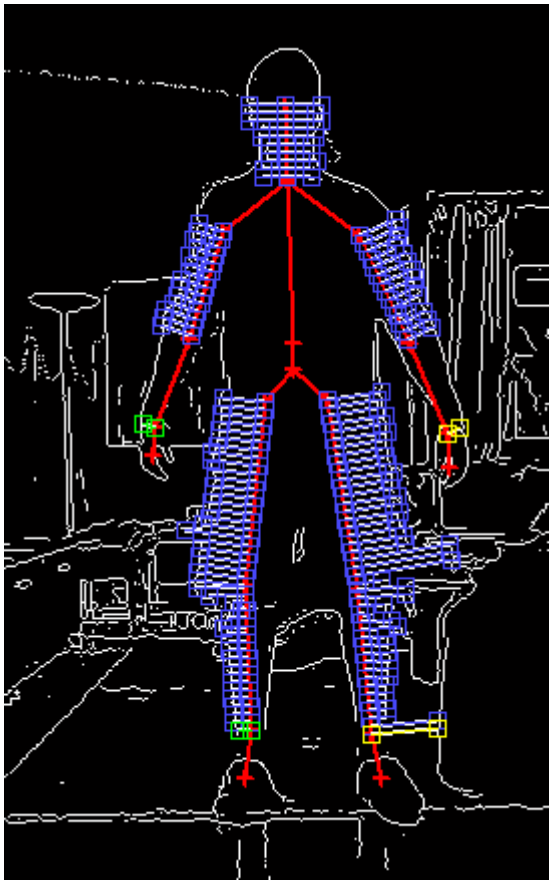


Figura 71: imagen procesada que devuelve MatLab con todas las funciones aplicadas.

Estas funciones devuelven los mismos datos:

- *El radio de la anchura que vamos a medir en píxeles.*
- *La distancia de las dos articulaciones en píxeles.*
- *La distancia en centímetros de las dos articulaciones.*

Estas funciones tendrán como parámetros de entrada:

- La imagen que vamos a procesar.
- JointImageIndices: los índices de las articulaciones respecto a la imagen en RGB.
- JointWorldCoordinates: los índices de las articulaciones con respecto al dispositivo Kinect.
- El primer índice de la articulación: un extremo de recta con la que trabajamos.
- El segundo índice de la articulación: otro extremo de recta con la que trabajamos.

La función anchura (anchura1, anchura2, anchura3, anchura4) realiza prácticamente el mismo proceso:

1. Dado que conocemos los índices de las articulaciones, utilizamos la función de medir la distancia entre píxeles llamada imdistline. Esta función crea una herramienta llamada Distance Tool (herramienta de distancia) sobre la distancia entre dos puntos que acabamos de medir.

   La función devuelve un identificador de un objeto imdistline. La herramienta de distancia es una línea de tamaño y posición variable que puede ser modificada por el ratón o por el teclado. Tiene la capacidad de medir la distancia entre dos puntos o píxeles. La

herramienta de distancia por defecto muestra la distancia en una etiqueta de texto superpuesto sobre la línea.

2. La herramienta de distancia contiene una estructura con funciones, llamada API, que se puede utilizar para recuperar información sobre la distancia y controlar otros aspectos de la herramienta de la distancia. Para acceder a esta estructura, utilizamos la función ipgetapi.

3. Calculamos la distancia en píxeles entre las dos articulaciones.

4. Calculamos la distancia en centímetros entre las dos articulaciones.

```
hline = imdistline(gca,[joI(num1,1)    joI(num2,1)],[joI(num1,2) joI(num2,2)]);

api = iptgetapi(hline);

medida_pixel = api.getDistance();

medida_real =sqrt(((joC(num1,1)-(joC(num2,1)))^2)+(joC(num1,2)-(joC(num2,2)))^2);
```

Código 12: cálculo de distancia en centímetros entre dos articulaciones (num1 y num2).

5. La imagen tiene 640x480 píxeles y se puede utilizar como si fuese un plano cartesiano. El punto de origen seria el punto (1,1) y estaría en la esquina superior izquierda de la imagen. Una coordenada va de 1 hasta 640 y el valor del índice es mayor mientras más a la derecha de la imagen se encuentre y la otra coordenada desde 1 hasta 480 y el valor del índice es mayor cuanto más debajo de la imagen se encuentre.

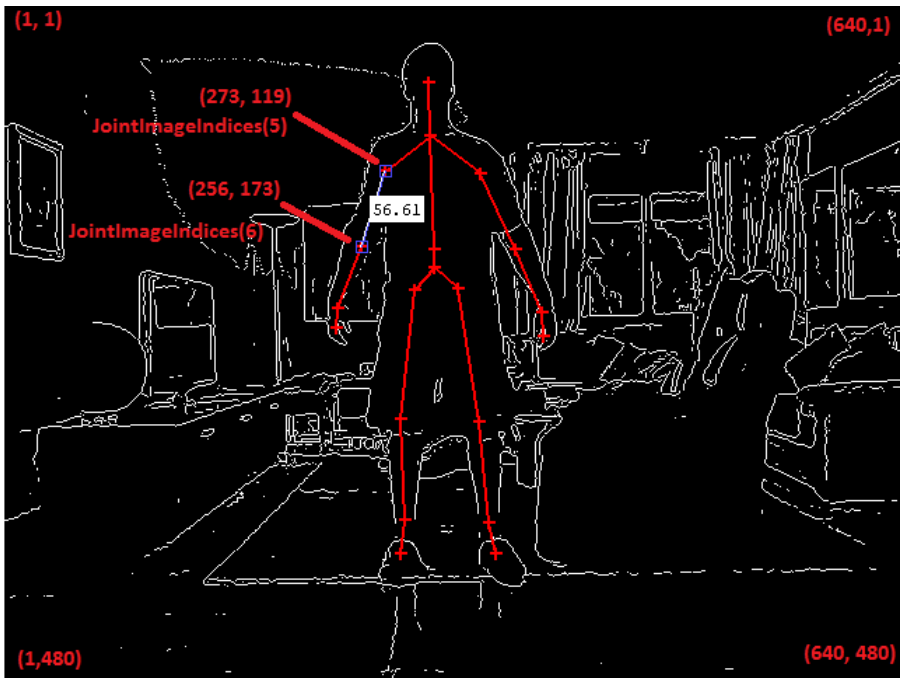


Figura 72: imagen de bordes con articulaciones y distancia en píxeles entre las articulaciones 5 y 6.

Ahora es el turno de trabajar con las rectas. Como observamos en la imagen superior, tenemos una recta de color azul que la hemos calculado gracias a la herramienta de distancia. Utilizando la estructura de funciones api previamente activada, usando una de sus funciones calcularemos el ángulo (en grados) de la recta respecto al eje horizontal. Después borramos la línea creada ya que no será necesaria.

Ahora que sabemos cuál es el ángulo de la recta utilizando la función de cálculo de la tangente en grados (***tand***) de Matlab, sabemos cuál es la pendiente de la recta.

```
angle = api.getAngleFromHorizontal();

api.delete();

% Returns the angle in degrees between the line d
%The angle returned is between 0 and 180 degrees.

%calculo de la pendiente de la recta
m = tand(angle);
```

Código 13: cálculo de ángulo en grados y pendiente de la recta entre dos articulaciones.

6. Conocido el ángulo en grados de la recta que une las dos articulaciones en la imagen, calculamos la pendiente perpendicular sobre esa recta. La idea es ir calculando rectas perpendiculares respecto a la recta que une las dos articulaciones del extremo de la recta hasta el otro.

```
m_per=- (1/m);
```

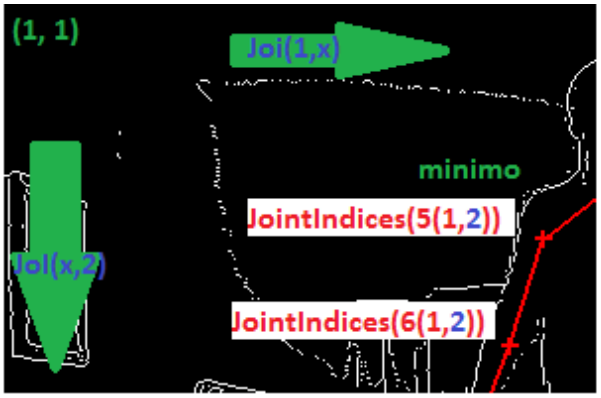
Código 14: cálculo de pendiente perpendicular.

7. Creamos una lista vacía para guardar las distancias de las rectas perpendiculares.

```
lista=[];
```

Código 15: creación de una lista vacía.

8. La primera recta perpendicular la situamos donde esté la menor coordenada Y de las dos articulaciones. Dicho de otra manera, la situamos en la coordenada más cercana al eje horizontal. Después, utilizando un *for*, iremos de cuatro píxeles en cuatro píxeles hasta llegar a la otra articulación.




```
if (joI(num1,2) < joI(num2,2))
    min_x=joI(num1,1);
    min_y=joI(num1,2);
    max_x=joI(num2,1);
    max_y=joI(num2,2);
else
    min_x=joI(num2,1);
    min_y=joI(num2,2);
    max_x=joI(num1,1);
    max_y=joI(num1,2);
end

for j=min_y:4:max_y,
```

Código 16: cálculo de la menor coordenada Y.

9. Dado que conocemos la pendiente de la recta (**m**) y la coordenada Y, podemos calcular la coordenada X de la siguiente manera. Os vamos a demostrar cómo se obtiene la fórmula.

Dado un punto (a,b) y una pendiente m

y-b = m (x -a) → *y-b = mx – ma* → ***y = mx – ma + b***

y-b = m (x -a) → *(y-b)/m = x –a* → ***x = ((y-b)/m) + a***

```
for j=min_y:4:max_y,

    i = round((-(j - min_y)/m) + min_x);
```

Código 17: cálculo de nuevo punto obtenido (i,j), el primer punto de la perpendicular.

10. Una vez que conocemos el primer punto de la perpendicular y la pendiente de la perpendicular (**m_per**) vamos hacia el borde (píxel con valor 1) hasta encontrarlo. Lo haremos de 3 píxeles en 3 píxeles hasta encontrar el borde. Ahora es cuando las funciones difieren: anchura1 y anchura3 irán hacia la izquierda, es decir, restarán tres píxeles a su coordenada X y las funciones anchura2 y anchura4 irán hacia la derecha, es decir, sumaran tres píxeles. Utilizando la formula previamente escrita, calculamos la coordenada Y.

```
j_per = round(-m_per * i_per + m_per * i + j);
```

Código 18: cálculo de coordenada Y conociendo pendiente (m_per) y coordenada X (i_per).

11. Dado que conocemos el punto de la perpendicular (**i_per, j_per**), debemos averiguar si ha llegado al borde. Para ello usamos una condición (**if**):
    - En caso de que entre sus 8 píxeles de alrededor haya un píxel de valor 1.
    - En caso de que la distancia recorrido en la perpendicular sea la mitad que la distancia entre las dos articulaciones (*esta condición la implantamos debido que a la imagen de los bordes obtenida no siempre es perfecta y a veces suele haber huecos*)

Se sigue buscando en el siguiente píxel de la perpendicular.

```
if((sum(sum(ima(j_per-1:j_per+1,i_per-1:i_per+1)))~=0)| (distancia_per > (medida_pixel/2)))
```

Código 19: búsqueda de píxel con valor 1.

He aquí una representación grafica de lo que queremos conseguir.

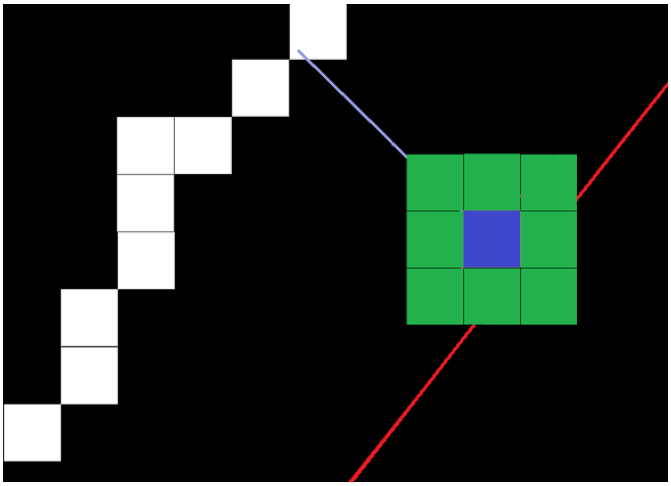


Figura 73: inicio de búsqueda de pixel con valor 1 (color blanco).

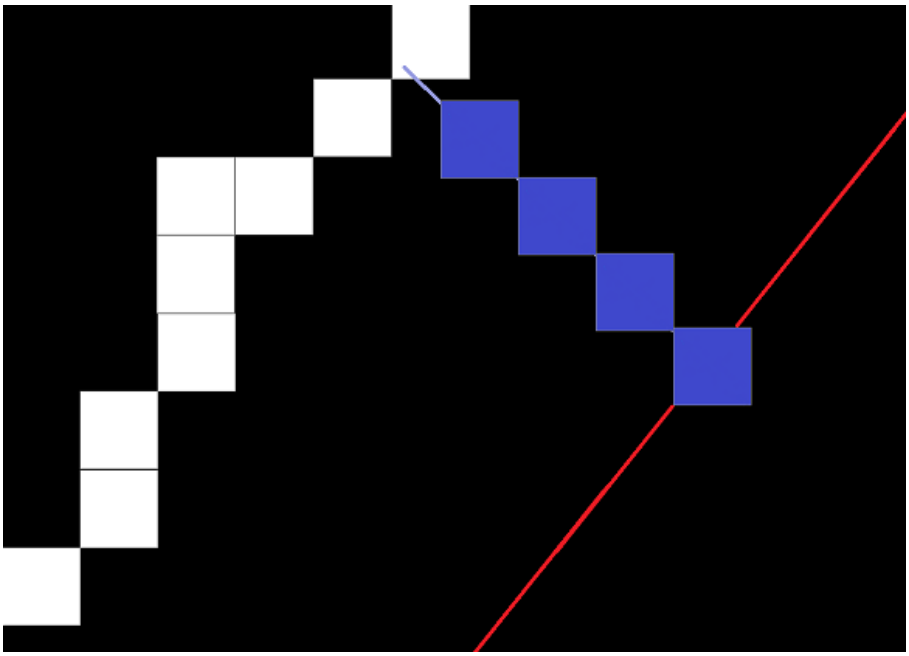


Figura 74: píxeles necesarios para la búsqueda del píxel con valor 1 (color blanco).

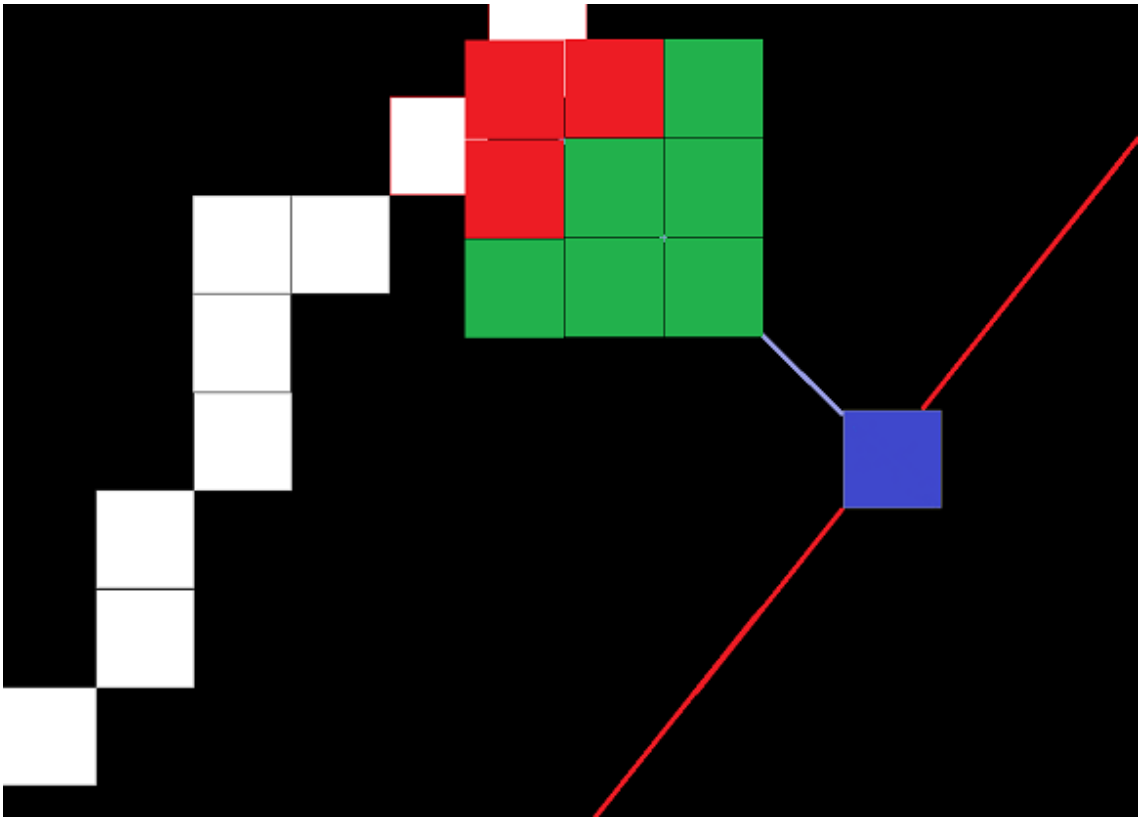


Figura 75: búsqueda de píxel con valor 1 (color blanco) finalizada.

Al finalizar la búsqueda de bordes, vamos a encontrarnos con la siguiente imagen en MatLab:

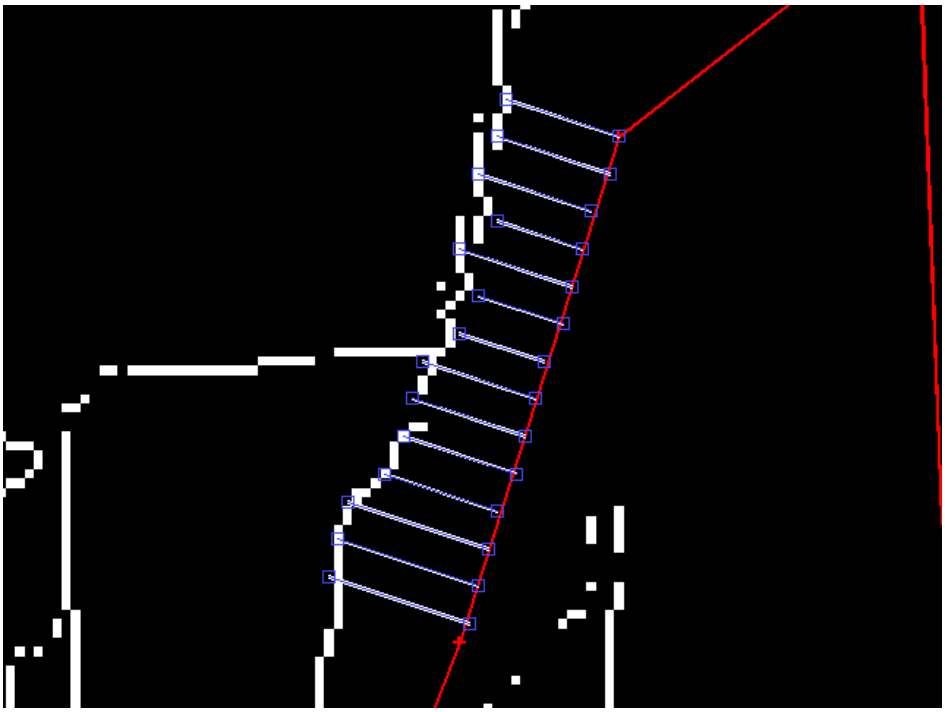


Figura 76: búsqueda de bordes entre dos articulaciones.

12. Cada distancia de las perpendiculares se irá introduciendo al final de la lista creada anteriormente.

```
lista= [lista distancia_per];
```

Código 20: inserción de distancia perpendicular en lista.

13. Una vez la lista ha sido completada, la ordenamos de menor a mayor utilizando la función de MatLab **sort**. Después de realizar la ordenación se eliminarán los valores grandes como por ejemplo, aquellas distancias que no han encontrado borde porque había un hueco y han seguido creciendo.

```
lista = sort(lista);
```

Código 21: ordenación de lista.

14. Después de ordenar la lista, nos quedamos con los valores de distancia más pequeños: con los de la mitad de la lista para abajo. Con esto conseguimos eliminar definitivamente los valores grandes.

```
lista = lista(1:round(length(lista)/2));
```

Código 22: recorte de lista con la primera mitad de elementos.

En la anchura del cuello, por ejemplo, se toman medidas que no son propias del cuello: Como parte de la cara y parte del pecho. Pero, ordenando de menor a mayor y cogiendo los valores más pequeños descartamos estas medidas que no son del cuello.

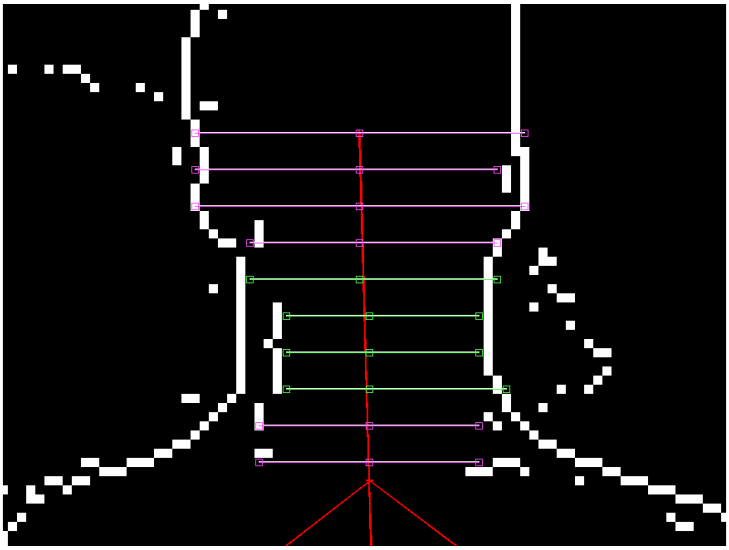


Figura 77: rectas utilizadas para el cálculo de los bordes del cuello.

15. A continuación calculamos la media de los menores valores de distancias obtenidos para hallar el radio de la anchura medida en píxeles que necesitamos.

```
radio = mean(lista);
```

Código 23: media de los elementos de la lista *lista*.

Al finalizar, la función devuelve los siguientes datos de salida:

1. El radio en píxeles de la anchura que vamos a medir.
2. La distancia en píxeles de las dos articulaciones.
3. La distancia en centímetros de las dos articulaciones.

Para la correcta comprensión de la siguiente explicación, vamos a nombrar los términos anteriores con nombres más característicos: radio_píxel (1), distancia_píxel (2) y distancia_real (3).

Para computar la anchura real en centímetros, vamos a aplicar la ***regla de tres simple directa***.

Esta regla se fundamenta en una relación de proporcionalidad.

$$\frac{\text{radio\_píxel}}{X} = \frac{\text{distancia\_píxel}}{\text{distancia\_real}}$$

Computamos X que es el radio de la anchura en centímetros.

$$x = \frac{\text{radio\_píxel}}{\text{distancia\_píxel}} * \text{distancia\_real}$$

Ahora que sabemos el radio de la anchura en centímetros aplicaremos la fórmula del cálculo del perímetro de la circunferencia.

$$anchura\_centimetros = radio\_centimetros * \pi * 2$$

De esta manera seremos capaces de calcular las 6 anchuras que necesitamos.

10.3.1.3. Calculo de medidas del torso

Una vez medidas las nueve longitudes y las seis anchuras, vamos a conseguir cuatro medidas que nos faltan:

- Contorno de pecho
- Contorno de bajo pecho
- Contorno de cintura
- Contorno de cadera.

Para ello conseguiremos los datos que nos proporcionan las tablas de medidas de la revista de moda **Burda Style**. Esta revista alemana fue fundada en 1949 por Aenne Burda, una publicista alemana símbolo del milagro económico alemán. Esta revista se publica en 16 idiomas y se distribuye por 89 países. Esta especializada en moda y en patrones de costura. Cada número de Burda Style contiene los patrones para cada diseño de prenda que anuncian, por lo que los lectores pueden crear la prenda que están en la revista. Un patrón de costura es una plantilla hecha en papel para ser copiada en el tejido y fabricar una prenda de vestir, cortando, armando y cosiendo las distintas piezas de la prenda. Existen dos tipos de patrones:

- Patrones domésticos: suelen ser de papel (normalmente papel seda), incluyen instrucciones de uso, sugerencias sobre la tela más apropiada para crear la prenda y sus posibles adaptaciones.

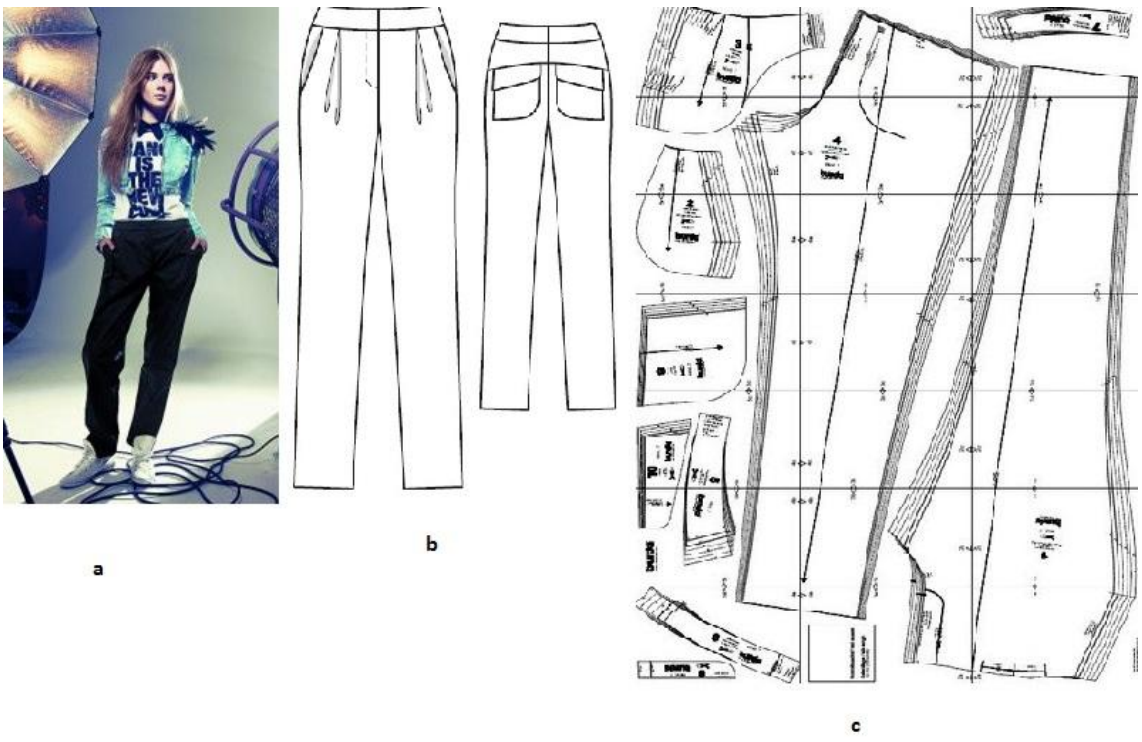


Figura 78: c) Patrón de costura del pantalón. b) Piezas del patrón unidas. a) Resultado final. (*)

(*) Fuente: http://www.elbauldelacosturera.com/

La figura 78 es el patrón utilizado para crear **pantalón Modelo 007 Burda Fácil** de línea recta con pinzas encontradas en la parte delantera y original bolsillos con tapas y canesú en la parte trasera. Tallas desde la 34 a la 44. Para descargarse el patrón y las instrucciones mirar en los enlaces externos del proyecto.

- Patrones industriales: Este tipo de patrones empieza con un borrador que se asemeja a la idea que ha tenido el diseñador de moda. El patrón se realiza en papel manila (parecido al papel seda pero más opaco), se aplica a un tejido de prueba y se confecciona la prenda. La prenda creada se prueba en un maniquí o un modelo para que el diseñador la apruebe.

  Una vez aprobado, se presenta a los clientes, generalmente a los clientes mayoristas. Cuando los clientes aprueban el potencial de la prenda, se realiza el escalado de la ropa. La exactitud de las tallas, los contornos y las líneas para coser se examinan, se corrigen los errores y se procede a su producción industrial para su posterior comercialización.

  En caso de que la prenda tenga éxito comercial, los patrones son guardados para su futuro uso.

Para crear la prenda usando los patrones la revista Burda contiene tablas de medidas. Estas tablas suelen ser diferentes respecto a sexo y edad. Hay tablas de mujer, hombre, niños y niña.

Hemos utilizado estas tablas para conseguir los cuatro datos que nos faltan.

Esta es la tabla que hemos utilizado en el caso que el usuario sea una mujer.

Talla	34	36	38	40	42	44	46	48	50	52
Contorno pecho (cm)	80	84	88	92	96	100	104	110	116	122
Contorno bajo pecho (cm)	66	70	74	78	82	86	90	94	98	102
Contorno cintura (cm)	62	66	70	74	78	82	86	92	98	104
Contorno caderas (cm)	86	90	94	98	102	106	110	116	122	128
Largo de espalda (cm)	41	41	42	42	43	43	44	44	45	46
Largo brazo (cm)	59	60	61	62	63	64	65	66	67	68
Contorno de cuello (cm)	34	35	36	37	38	39	40	41	42	43

Tabla 5: **tabla de medidas de costuras de mujeres con tallas diferentes.**

Esta es la tabla que hemos utilizado en el caso que el usuario sea un hombre

Talla	44	46	48	50	52	54	56
Contorno pecho (cm)	88	92	96	100	104	108	112
Contorno bajo pecho (cm)	81	85	89	93	97	101	105
Contorno cintura (cm)	78	82	86	90	94	98	102
Contorno caderas (cm)	90	94	98	102	106	110	115
Largo de espalda (cm)	42	43	43.5	44.5	45	45.5	46

Largo brazo (cm)	61	62	63	64	65	66	67
Contorno de cuello (cm)	37	38	39	40	41	42	43

Tabla 6: tabla de medidas de costuras de hombres con tallas diferentes.

Las medidas mayores y menores que las de la tabla se calculan fácilmente ya que la diferencia entre ellas es siempre la misma. Por ejemplo:

- Largo brazo: 61-62-63→ diferencia de 1. Por lo que los menores serán 59-58-57-56 y mayores 68-69-70-71
- Contorno de pecho: 88-92-96-100→ diferencia de 4. Por lo que los menores serán 84-80-76-72 y mayores 116-120-124-128.

De estas tablas nos interesan 5 datos:

- Los cuatros datos que desconocemos y que tenemos que encontrar
- El dato que conocemos: largo de brazo. En la moda, largo de brazo es la distancia que va desde el hombro hasta la muñeca. Los datos de los índices de estas dos articulaciones ya se conocen. Por lo que solo tenemos que computar los datos. El dato sale de la suma de la longitud del brazo superior y de longitud del brazo inferior o antebrazo.

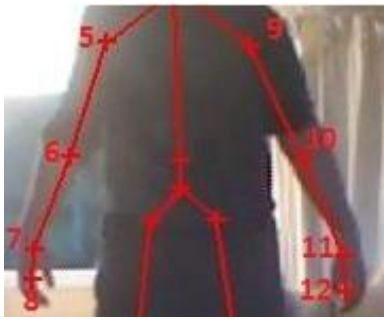


Figura 79: Imagen de articulaciones de brazo superior e inferior.

Una vez calculado con la Kinect el largo del brazo, Buscaremos en la tabla cuales son los cuatro datos que le corresponden. Por ejemplo, si el usuario es una mujer y tiene un largo de brazo de 59 centímetros, tendrá un contorno de pecho de 80 centímetros, un contorno de bajo pecho de 66 centímetros, un contorno de cintura de 62 centímetros y un contorno de cadera de 86 centímetros.

10.3.1.4. Guardar datos

Después de calcular las 19 medidas, procederemos a guardarlos en un fichero llamado *datos.txt* para su posterior traslado a Makehuman.

10.4. ANEXO D: Fase 4: Estudio de Base de datos de usuarios y prendas

10.4.1 Creación de Base de Datos.

Dado que conocemos las 19 medidas que necesitamos del usuario, para el registro de prendas vamos a utilizar patrones. Necesitamos conocer las medidas de estos patrones. Después solo habrá que introducir las medidas del usuario en las medidas de los patrones para que la prenda encaje a la perfección y no necesite devolverla.

Estos son los datos que necesitamos guardar de cada usuario en la base de datos:

- nombre
- identificador
- Datos personales (dirección, teléfono, e-mail…)
- 19 medidas captadas por kinect

En el caso de las prendas necesitamos saber los siguientes parámetros:

- Pantalón:
  - anchura cadera
  - anchura pierna superior
  - longitud pierna superior
  - longitud pierna inferior

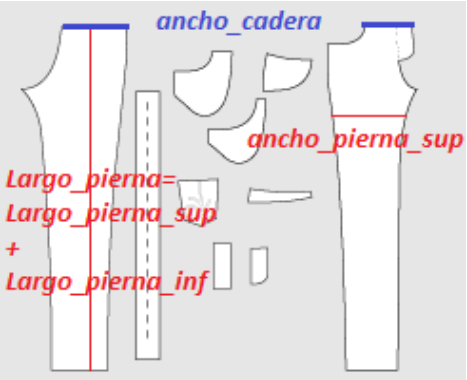


Figura 80: patrón de costura de pantalón con los parámetros necesarios para ajustar. (*)

(*) Fuente: http://angelakane.com/

- Camiseta:
  - anchura cuello
  - anchura brazo superior
  - anchura pecho
  - distancia desde nuca a espalda

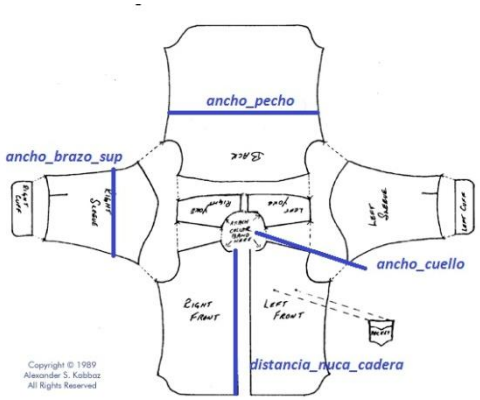


Figura 81: patrón de costura de camiseta con los parámetros necesarios para ajustar (*)

(*) Fuente: http://www.molendrix.com/

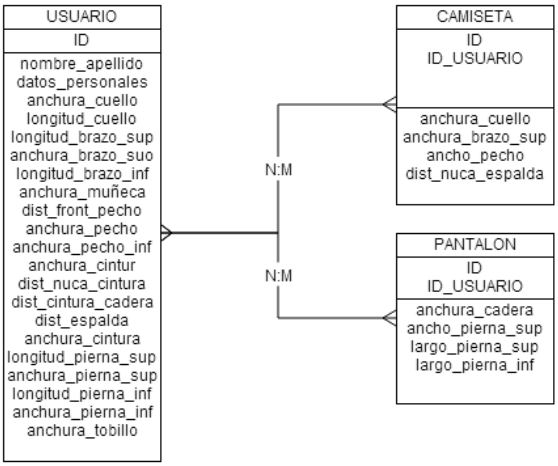


Figura 82: entidades de usuario, camiseta y pantalón.

10.4.2. Nuevo sistema de comercio electrónico

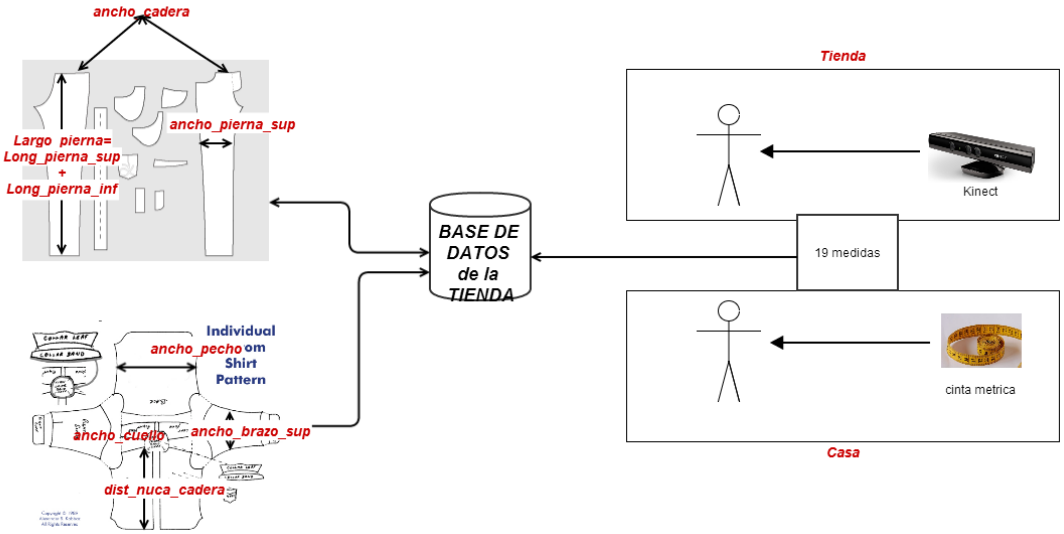


Figura 83: diagrama del nuevo sistema de comercio electrónico de compra venta de ropa.

10.5. ANEXO E: Cargar maniquí con las medidas de usuario

10.5.1. Inserción de datos.

Una vez hemos conseguido y hemos guardado las 19 medias, es hora de crear el maniquí. Antes de empezar a introducir las medidas, tenemos que configurar el género, la altura y el peso del usuario. Esto lo haremos en la pestaña principal de Makehuman. La primera que aparece. Está en el apartado ***Macro***, dentro de ***Modeling***.

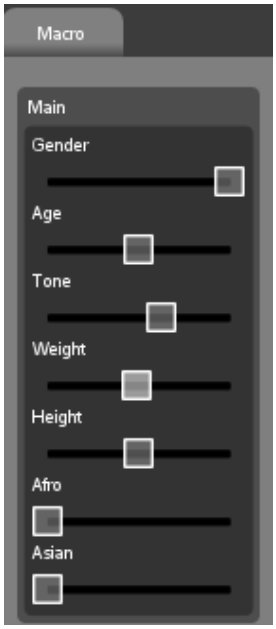


Figura 84: Cuadro de modelado principal del modelo.

En este apartado es posible cambiar:

1. ***Género (Gender):*** Si la barra deslizadora está a la derecha el género del maniquí será masculino y si esta a la izquierda será femenino. Si está en el medio será una mezcla de los dos géneros.
2. ***Edad (Age):*** Si la barra deslizadora está en la derecha la edad del maniquí de 70 años y si está a la izquierda la edad será 12 años. Es decir, podemos definir edades comprendidas entre los 12 y los 70 años.
3. ***Tono muscular (Tone)***. Si la barra deslizadora está en la izquierda el maniquí no tendrá musculatura y si está a la derecha tendrá mucha musculatura.
4. ***Peso (Weight)***: En caso de que la barra esté en la izquierda, el maniquí tendrá un peso bajo y por lo tanto será un maniquí delgado. En caso de que la barra esté en la derecha, el maniquí sobrepeso y será un maniquí gordo. Si está en medio tendrá un peso medio.
5. ***Altura (Height)***: Si la barra deslizadora está en la derecha la mayor altura que podemos configurar es de 244 centímetros y si está a la izquierda la menor edad que podemos configurar es de 147 centímetros.
6. ***Afro y Asian***: Es posible dar rasgos asiáticos y africanos en el maniquí. La propiedad de la altura varía dependiendo de estos rasgos. Los maniquís africanos tienen mayor altura que maniquís no africanos y los maniquís con rasgos asiáticos tienen menor altura que los maniquís sin estos rasgos.

Figura 85: ejemplos de rasgos asiáticos y africanos. (*)

(*) Fuente: http://makehuman.org

Este es el ejemplo de unas medidas recogidas por Kinect.

```
neck circum: 35.998578
neck height: 9.552225
upperarm circum: 37.167962
upperarm lenght: 29.549870
lowerarm lenght: 25.630654
wrist circum: 23.237100
front chest dist: 35.471140
bust circum: 116.000000
underbust circum: 109.000000
waist circum: 106.000000
hips circum: 120.000000
nape to waist dist: 41.772482
waist to hip dist: 7.760736
shoulder dist: 22.138975
upperleg height: 48.783755
thigh circum: 72.134493
lowerleg height: 39.381933
calf circum: 29.892218
ankle circum: 26.807778
```

Figura 86: ejemplos de medidas capturadas por Kinect.

Así quedaría configurado el maniquí después de introducir las 19 medidas en los campos correspondientes del apartado Measure.

Se ha intentado introducir las medidas a través del código del Makehuman. Para ello se necesita instalar *command line* [18]. No se ha conseguido debido a incompatibilidades con las versiones de Python de command line y el python que utiliza Makehuman. Esta tarea queda por desarrollar.

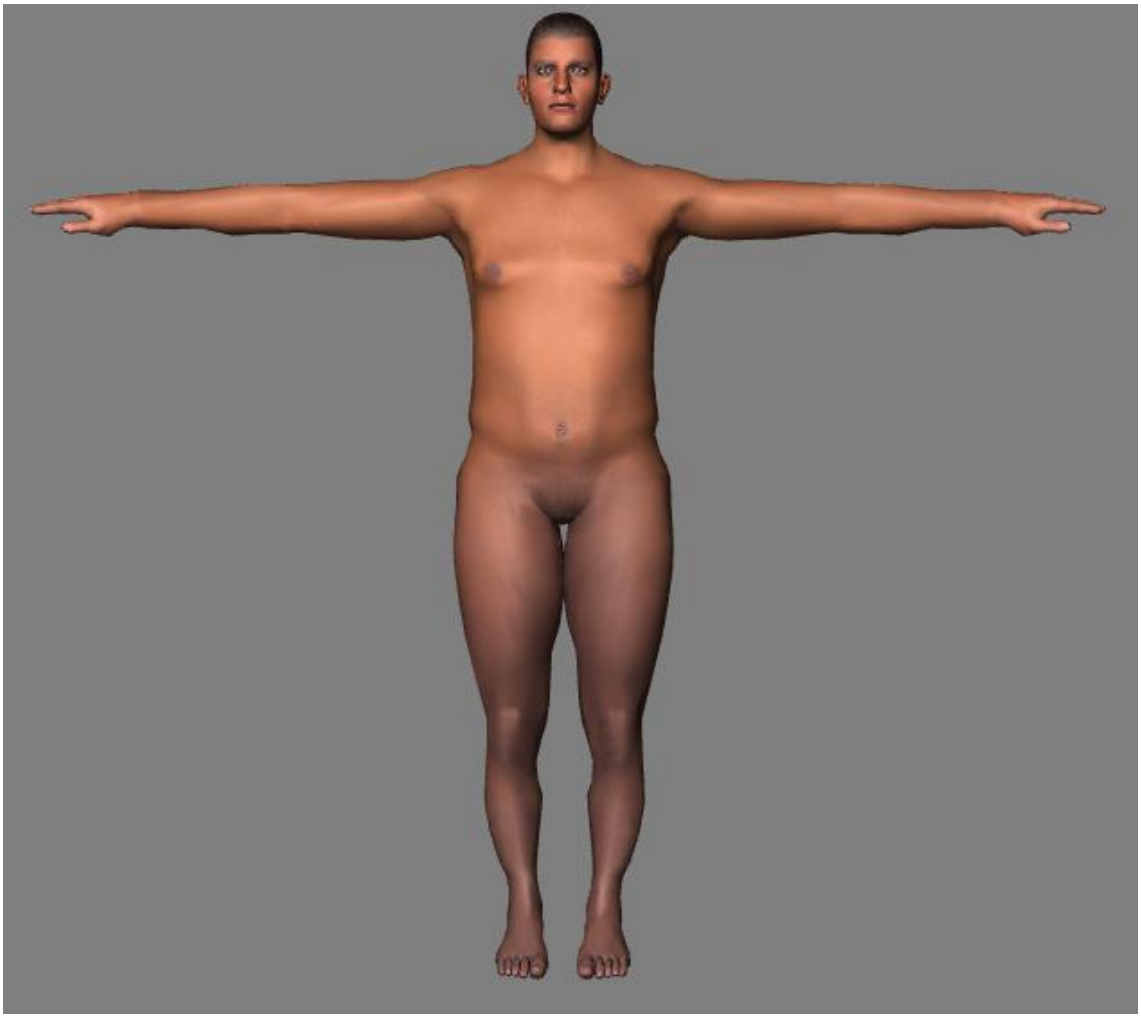


Figura 87: modelo de Makehuman con las medidas recogidas por Kinect.

10.6. ANEXO F: BLENDER

10.6.1. Introducción a Blender

Queremos crear ropa realista en tres dimensiones, es decir, con la imagen frontal y trasera de una prenda de vestir conseguir que tenga forma tridimensional. Para ello aparte de las dos imágenes necesitamos un programa informático que lo consiga.

Hoy en día existen muchos programas informáticos para crear animaciones en tres dimensiones. Estos son alguno de ellos:

- Allegorithmic
- Autodesk 3ds Max
- Autodesk Maya
- ZBrush
- Cinema 4D
- Blender

Al estar utilizando Makehuman necesitamos un programa que sea compatible con él. El único que es compatible es Blender. Los dos programas utilizan Python.

BLENDER

Blender es un programa informático multiplataforma (GNU/LINUX, Mac/OS X, Windows, IRIX, Solaris y FreeBSD) y gratuito dedicado al modelado, iluminación, "renderizado", simulación de líquidos y humo, edición de videos, animación y creación de gráficos tridimensionales. También es posible desarrollar videojuegos gracias a su motor de juegos interno.

Estos son los requisitos de Hardware de Blender:

Hardware	Mínimo	Recomendado	Óptimo
Procesador	2 GHz Doble núcleo de 32-bit	Cuádruple núcleo de 64-bit	Óctuple núcleo de 64-bit
Memoria	2 GB RAM	8 GB RAM	16 GB RAM
Tarjeta grafica	Tarjeta OPENGL con 256 MB video RAM	Tarjeta OPENGL con 1 GB de video RAM	Tarjeta de Doble OPENGL con e GB de video RAM
Resolución	1280x786 pixeles, 24 bit color	1920x1080 pixeles, 24 bit color	Dual 1920x1080 pixeles, 24 bit color
Entrada	Ratón de dos botones	Ratón de tres botones	Ratón de tres botones y tarjeta digitalizadora

Tabla 7: requisitos de Hardware de Blender.

Existen muchas versiones de Blender. He aquí un breve historial sobre las versiones:

1. **Rama 1.0 (1.0, .1.23, 1.3, 1.4, 1.5, 1.6, 1.8)**
   Desde enero de 1995 hasta agosto de 2000. Blender comenzó a desarrollarse en el estudio de animación NeoGeo en Holanda. En esta etapa se desarrollan las primeras versiones comerciales, hasta que en junio de 2000, pasa a ser software libre. Se desarrollan versiones para otros sistemas operativos.
2. **Rama 2.0 (2.00, 2.10)**

Desde agosto de 2000 hasta agosto de 2001. Se modifica al lenguaje de programación Python.

3. **Rama 2.2 (2.20, 2.21, 2.22, 2.23, 2.24, 2.25, 2.26,2.27, 2.28)**

   Desde agosto de 2001 hasta octubre de 2003. En esta etapa se funda Blender Fundation, una fundación encargada de continuar el desarrollo de Blender y promover su uso. Blender se convierte en código abierto y se imparte la primera conferencia de Blender.

4. **Rama 2.3 (2.30, 2.31, 2.32, 2.33, 2.34, 2.35, 2.36, 2.37)**

   Desde octubre de 2003 hasta diciembre de 2005. Se introducen las herramientas de transformación.

5. **Rama 2.4 (2.40, 2.41, 2.42, 2.43, 2.44, 2.45, 2.46, 2.47, 2.48, 2.48, 2.49, 2.49a, 2.49b)**

   Desde diciembre de 2005 hasta diciembre de 2009. En verano de 2007 se funda el instituto Blender, un instituto encargado de proyecto de animación. Un año más tarde se presenta el cortometraje *Big Buck Bunny* y el videojuego *Yo Frankie*!

Figura 88: carteles de Big Buck Bunny y Yo Frankie. (*)

(*) Fuente: http://www.blender.org/

6. **Rama 2.5 (2.50 Alpha 0, 2.50 Alpha 1, 2.50 Alpha 2, 2.53 Beta, 2.54 Beta, 2.55 Beta, 2.56 Beta, 2.56a Beta, 2.57 Stable, 2.58 Stable, 2.58a Stable, 2.59 Stable)**

   Desde diciembre de 2009 hasta agosto de 2011. Se presenta el cortometraje Sintel.

7. **Rama 2.6 (2.60, 2.61, 2.62, 2.63, 2.64, 2.65, 2.65a, 2.66, 2.67, 2.67ª, 2.67b, 2.68, 2.68a, 2.69)**

   Desde agosto de 2011 hasta marzo de 2014. Se presenta el cortometraje Tears of Steel.

8. **Rama 2.7 (2.70)**

   Desde marzo de 2014 hasta ahora. Comienza el primer largometraje producido por Blender Fundation, llamado Gooseberry. Costó 18 meses en completarse y la campaña se financió a través del *cloudfounding*. Esta financiación se basa una suscripción mensual al proyecto y a cambio se recibe acceso a todos los recursos digitales que tiene el proyecto de Blender. Debido al acceso a los recursos, se puede verificar todo el proceso.

Blender se actualiza constantemente por medio de desarrolladores, este es el motivo de que existan tantas versiones [18].

Blender posee una interfaz muy completa. Esta interfaz se dibuja en OpenGL [17], es posible organizar la interfaz de la forma que se quiera. La organización puede ser guardada. Para acelerar el

trabajo existen atajos del teclado. Los atajos también pueden ser modificados y guardados para facilitar su recordatorio.

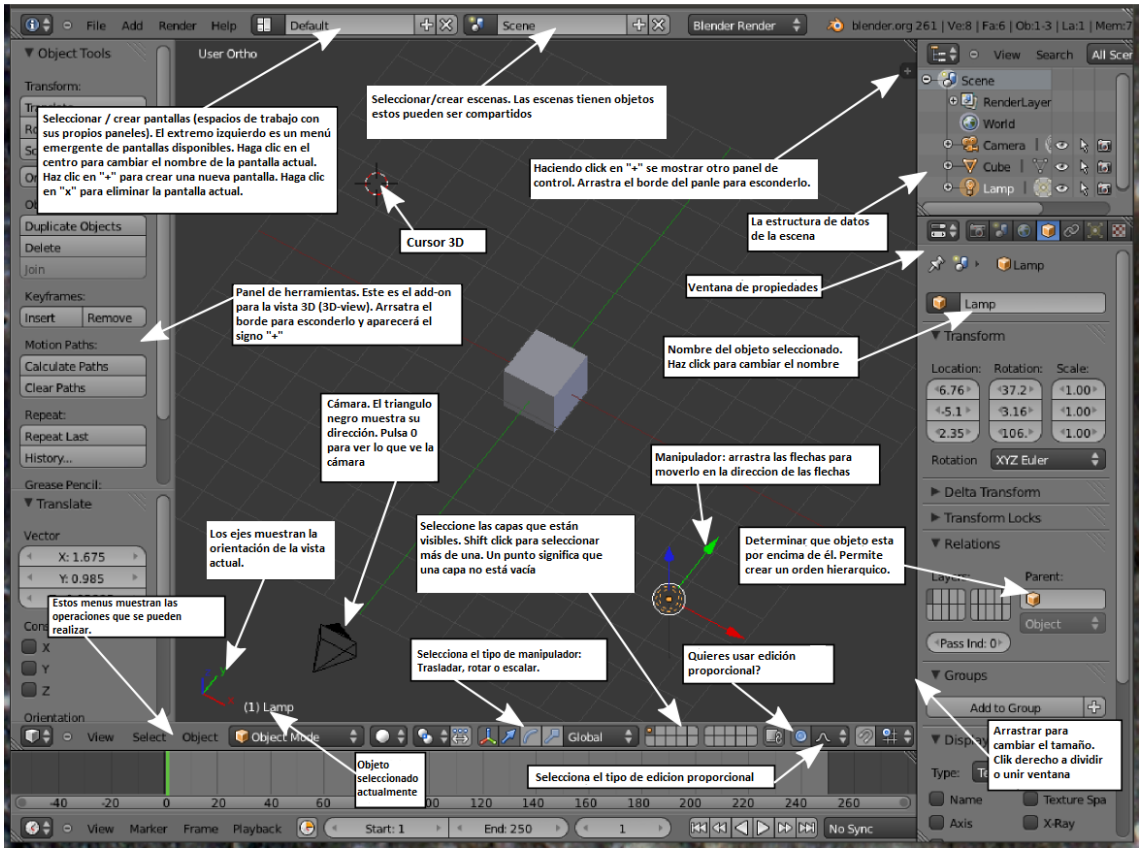


Figura 89: paneles de control de Blender.

10.7. ANEXO G: Detectar el dispositivo Kinect en MatLab

10.7.1. Desarrollo

Normalmente en *Image Adquisition Toolbox*, [8] cada imagen o cámara del dispositivo tiene un DeviceID. Debido a que Kinect tiene dos cámaras separadas, la cámara RGB y la cámara de profundidad, la toolbox define dos DeviceID. Lo podemos comprobar en el siguiente ejemplo:

```
Command Window
>> info = imaqhwinfo('kinect')

info =

       AdaptorDllName: [1x79 char]
    AdaptorDllVersion: '4.5 (R2013a)'
          AdaptorName: 'kinect'
            DeviceIDs: {[1]  [2]}
           DeviceInfo: [1x2 struct]

>> info.DeviceInfo(1)
```

Código 24: detección de Kinect utilizando MatLab.

Se puede observar que existen dos DeviceID. DeviceID →1 es la cámara de color o cámara RGB, mientras que DeviceID→2 es la cámara de profundidad.

```
>> info.DeviceInfo(1)

ans =

             DefaultFormat: 'RGB_640x480'
       DeviceFileSupported: 0
                DeviceName: 'Kinect Color Sensor'
                  DeviceID: 1
     VideoInputConstructor: 'videoinput('kinect', 1)'
    VideoDeviceConstructor: 'imaq.VideoDevice('kinect', 1)'
          SupportedFormats: {1x4 cell}

>> info.DeviceInfo(2)

ans =

             DefaultFormat: 'Depth_640x480'
       DeviceFileSupported: 0
                DeviceName: 'Kinect Depth Sensor'
                  DeviceID: 2
     VideoInputConstructor: 'videoinput('kinect', 2)'
    VideoDeviceConstructor: 'imaq.VideoDevice('kinect', 2)'
          SupportedFormats: {'Depth_320x240'  'Depth_640x480'  'Depth_80x60'}
```

Código 25: detección de cámaras de color y profundidad de Kinect utilizando MatLab.

Como se puede apreciar, la cámara de color tiene una resolución por defecto de 640x480 en formato RGB y una frecuencia de 30 imágenes por segundo, y la cámara de profundidad tiene la misma frecuencia de imágenes y la misma cantidad de píxeles. Se pueden utilizar múltiples Kinects a la vez. Por

ejemplo, si tenemos dos Kinects, la cámara RGB de la primera Kinect tendría DeviceID→1 y su cámara de profundidad DeviceID→2, la cámara RGB de la segunda Kinect seria DeviceID→3 y su cámara de profundidad DeviceID→4.

Para crear las dos entradas de video, utilizamos DeviceID→1 para la entrada de color y DeviceID→2 para la entrada de profundidad. En caso de querer ver lo que está captando el video, utilizamos el comando *preview*.

```
colorVid = videoinput('kinect',1);
depthVid = videoinput('kinect',2);
preview(colorVid)
preview(depthVid)
```

Código 26: vista de video de color de color y profundidad generados por Kinect.

Para acceder a las propiedades específicas del dispositivo, debemos acceder a las propiedades del sensor de color de Kinect. En este proyecto, estamos usando la versión R2013a o también conocida como Matlab 8.1, la cual no utiliza todas las ventajas que nos da Kinect en MatLab. Este problema también aparecerá más adelante. En este caso, al ver las propiedades de la cámara de color, observamos lo siguiente:

```
>> colorVid = videoinput('kinect',1);
src = getselectedsource(colorVid)

   Display Summary for Video Source Object:

      General Settings:
        Parent = [1x1 videoinput]
        Selected = on
        SourceName = Color Source
        Tag =
        Type = videosource

      Device Specific Properties:
        CameraElevationAngle = 3
        FrameRate = 30
```

Código 27: Propiedades de la cámara de color en Matlab R2013a.

Podemos cambiar dos cosas:

- **CameraElevationAngle**. El ángulo del dispositivo Kinect. El valor introducido debe ser un número entero y debe estar entre 27 y -27. Esta propiedad también es compartida con la cámara de profundidad. Por ejemplo: src.CameraElevationAngle = 10.
- **FrameRate**. La cantidad de imágenes por segundo. Esta es una propiedad de lectura (*read-only*), es decir, no se puede modificar. Los valores posibles que puede dar la cámara de color son 12, 15 y 30 (por defecto).

En la versión de Matlab R2014a o MatLab 8.3, se puede cambiar muchas más propiedades.

Propiedad	Descripción
Accelerometer(Acelerómetro)	Devuelve un vector de tres dimensiones con los datos de aceleración para sensores tanto en el color como profundidad
AutoExposure (Auto-Exposición)	Se utiliza para ajustar la exposición (cantidad de luz que recibe) de forma automática. Valores: on/off
AutoWhiteBalance (Balance de blancos automático)	Se utiliza para ajustarel balance de blancos de forma automática. Esta relacionado con la propiedad WhiteBalance.
Backlightcompensation (compensación de contraluz)	Configura los modos de compensación de luz de fondo para ajustar la cámara para capturar imágenes que dependen de las condiciones ambientales. Valores: averagebrightness (brillo normal), centerPriority (prioridad centro de la escena), LowLightsPriority (prioridad en luces bajas) y CenterOnly (prioridad solo en el centro de la escena)
Brightness (Brillo)	Indica el nivel de brillo. Valores entre 0.0 y 1.0. Por defecto: 0.2156
Contrast (Contraste)	Indica el nivel de contraste. Valores entre 0.5 y 2. Por defecto: 1
ExposureTime (Tiempo de exposición)	Indica el tiempo de exposición en incrementos de 1/10.000 de un segundo. Valores entre 0 y 400. Por defecto: 0
FrameInterval (Intervalo de frecuencia)	Indica el intervalo de frecuencia en incrementos de 1/10.000 de un segundo. Valores entre 0 y 400. Por defecto: 0
Gain (Ganancia)	Indica el multiplicador para los valores de colores RGB. Valores entre 1.0 y 16.0. Por defecto: 1.0
Gamma	Indica la medición gamma. Valores entre 1 y 2.8. Por defecto: 2.2
Hue (Matiz)	Indica la configuración del tono o matiz. Valores entre -22 y 22. Por defecto: 0
	Opción para reducir el parpadeo (ruido) causado por la frecuencia de una voltaje exterior (ruido). Valores: Disabled (deshabilitado), fiftyHertz (50 Hercios) o sixtyHertz (sesenta Hercios)
Saturation (Saturación)	Indica el nivel de saturación. Valores entre 0 y 2. Por defecto: 1
Sharpness (nitidez)	Indica el nivel de nitidez. Valores entre 0 y 1. Por defecto: 0.5
WhiteBalance (Balance de blancos)	Indica la temperatura de color en grados Kelvin. Valores entre 2700 y 6500. Por defecto: 2700

Tabla 8: Propiedades de la cámara de color utilizando MatLab R2014.

También podemos acceder a las propiedades del sensor de profundidad de Kinect. A continuación mostramos las propiedades de la versión R2013a.

```
>> depthVid = videoinput('kinect',2);
src = getselectedsource(depthVid)

   Display Summary for Video Source Object:

      General Settings:
        Parent = [1x1 videoinput]
        Selected = on
        SourceName = Depth Source
        Tag =
        Type = videosource

      Device Specific Properties:
        BodyPosture = Standing
        CameraElevationAngle = 3
        DepthMode = Default
        FrameRate = 30
        SkeletonsToTrack = [1x0 double]
        TrackingMode = Off
```

Código 28: propiedades de la cámara de profundidad en Matlab R2013a.

Podemos observar las siguientes propiedades, algunas de ellas relacionadas con el seguimiento del esqueleto:

- **Bodyposture** (postura del cuerpo): Indica si el esqueleto que sigue la cámara está de pie o sentado. Si está de pie (*Standing*) devuelve 20 puntos del esqueleto y si está sentado (*Seated*) 10 puntos.

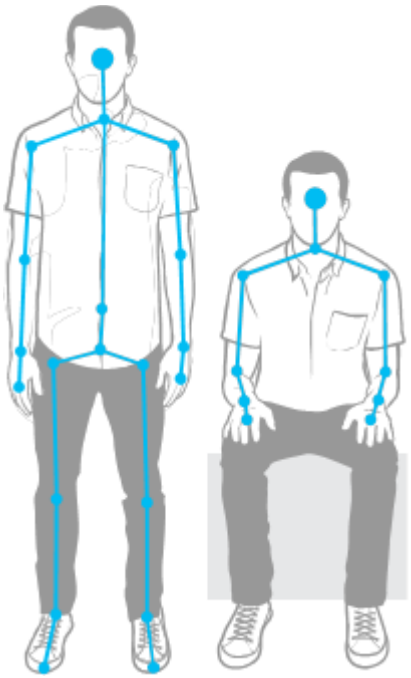


Figura 90: Ejemplos de articulaciones capturadas en dos posturas diferentes. (*)

(*) Fuente: http://iteches.com/

- **CameraElevationAngle**. El ángulo del dispositivo Kinect. Esta propiedad también es compartida con la cámara de color.
- **DepthMode** (Modo de profundidad): Indica el rango de profundidad en el mapa de profundidad. Valores Default (50 cm a 400 cm) o Near (40 cm a 300 cm)
- **FrameRate**. La cantidad de imágenes por segundo. Esta es una propiedad de lectura (read-only) y su valor es fijo: 30.
- **SkeletonsToTrack**. Indica el esqueleto Tracking ID devuelto como parte de Meta-Data (explicado más adelante). Valores: [] (por defecto), [TrackingID1] sigue 1 esqueleto con tracking ID = TrackingID1 y [TracckingID1 TrackingID2] sigue dos esqueletos con tracking ID = TrackingID1 y TrackingID2.
- **TrackingMode**. Indica el estado del seguimiento. Valores: *Skeleton* sigue un esqueleto completo con *joints* (articulaciones, explicado más adelante), *Position* sigue la posición de la cadera solamente y *Off*, por defecto, deshabilita el seguimiento.

En la versión de Matlab R2014a o MatLab 8.3, se puede cambiar muchas más propiedades.

Accelerometer(Acelerómetro)	**Devuelve un vector de tres dimensiones con los datos de aceleración para sensores tanto en el color como profundidad**
IREmiter (emisor de infrarrojos)	Controla el estado del emisor de infrarrojos. Valores: on/off

Tabla 9: Propiedades de la cámara de profundidad utilizando MatLab R2014.

Es posible cambiar el número de frames o fotogramas que se van a recibir por las dos cámaras. En este proyecto cambiamos esta propiedad para que obtenga 100 frames. Se ha escogido esta cantidad para darle a Kinect for Windows suficiente cantidad para que puede seguir el esqueleto.

```
triggerconfig([colorVid depthVid],'manual');
set([colorVid depthVid], 'FramesPerTrigger', 100);
```

Código 29: cambio de número de fotogramas capturados.

Gracias al comando *start*, empezamos a coger imágenes por las cámaras de color y profundidad. Va a comenzar la adquisición de datos (imágenes), pero no se va a iniciar la carga de las imágenes recibidas. Con el comando trigger iniciamos las cámaras para que inicien el registro de datos (imágenes).

```
start([colorVid depthVid]);
trigger([colorVid depthVid]);
```

Código 30: Inicio de captura de fotogramas.

Después recuperamos los datos que hemos adquirido y a continuación paramos la recepción de datos.

```
[colorFrameData, colorTimeData, colorMetaData] = getdata(colorVid);
[depthFrameData, depthTimeData, depthMetaData] = getdata(depthVid);

stop([colorVid depthVid]);
```

Código 31: recuperación de datos capturados por las dos cámaras y parada de captura de fotogramas.

La función [Datos, Tiempo, Meta-Datos]=***getdata****(video)* en caso de devolver:

- Un dato devuelve el número de fotograma especificado por FramesPerTrigger.
  data = getdata(video)
- Dos datos devuelve el número de fotogramas y el tiempo que ha tardado desde que se inició a tomar hasta cada fotograma.
  [data, tiempo] = getdata(video)
- Tres datos. devuelve el número de fotogramas, el tiempo que ha tardado desde que se inició a tomar hasta cada fotograma y la meta información que contiene cada fotograma.
- [data, tiempo, metadata] = getdata(video)

Tiene como propósito recoger tres tipos de datos:

- Datos, en nuestros caso:
  - colorFrameData: Datos RGB de todos los píxeles de los 100 fotogramas recogidos. Al haber recogido 100 fotogramas, cada fotograma tiene 640x480 píxeles y un píxel tiene 3 valores RGB, entonces su valor máximo será:
    100x640x480x3 = 92160000.

    Lo podemos comprobar de la siguiente manera

```
>> colorFrameData(92160000)

ans =

   44

>> colorFrameData(92160001)
Index exceeds matrix dimensions.
```

Código 32: comprobación de cantidad de información capturada por imagen de color.

  - DepthMetaData: Datos de profundidad de los píxeles emitidos por el emisor de infrarrojos, es decir, la distancia en milímetros que entre la cámara y los puntos emitidos por el emisor. Al haber recogidos 100 fotogramas, cada fotograma tiene 640x480 píxeles, entonces se valor máximo será:
    100x640x480 = 30720000

    Lo podemos comprobar de la siguiente manera:

```
>> depthFrameData(30720000)

ans =

   1920

>> depthFrameData(30720001)
Index exceeds matrix dimensions.
```

Código 33: comprobación de cantidad de información capturada por imagen de profundidad.

- Tiempo, una matriz 100 x 1 (un vector, 100 es el numero de fotogramas que hemos recogido) donde cada elemento del vector indica el tiempo relativo, en segundos, de la trama correspondiente en los datos, con relación al primer fotograma recibido.

  Las cámaras por defecto no están sincronizadas, es decir, las dos cámaras recogen datos en diferentes tiempos.

  En el ejemplo inferior, hemos hecho una captura de imagen. Hemos capturado las primeras 27 unidades de tiempo para que se puede comprobar la falta de sincronización.

```
colorTimeData =  depthTimeData =

    0.0146          0.7908
    0.0467          0.8213
    0.0830          0.8550
    0.1252          0.8855
    0.1502          0.9262
    0.1830          0.9539
    0.2154          0.9875
    0.2480          1.0187
    0.2798          1.1097
    0.3182          1.2623
    0.3475          1.2908
    0.3800          1.3904
    0.4155          1.4486
    0.4620          1.5752
    0.4794          1.6988
    0.5151          1.7399
    0.5480          1.7745
    0.5801          1.8859
    0.6153          1.9379
    0.6499          1.9887
    0.6824          2.0078
    0.7152          2.0861
    0.7473          2.1320
    0.7835          2.1649
    0.8169          2.2093
    0.8494          2.3231
    0.8821          2.3736
```

Código 34: comprobación de falta de sincronización de las dos cámaras.

- Meta-Datos, un vector de 100 estructuras de datos. Cada estructura contiene información sobre el fotograma correspondiente. En nuestro caso:
    - colorMetaData:

```
colorMetaData =

100x1 struct array with fields:

    AbsTime
    FrameNumber
    RelativeFrame
    TriggerIndex
```

Código 35: Propiedades de colorMetaData en Matlab R2013a.

He aquí una breve explicación de sus campos:

Campo	Descripción
AbsTime	El tiempo absoluto en el que fotograma fue adquirido. Representa la marca del tiempo completo, incluyendo fecha y hora, en formato de reloj de MatLab
FrameNumber	Número que identifica el enésimo fotograma, desde que se ordeno el comando *start*
RelativeFrame	Número que identifica el enésimo fotograma, desde que se inicio a adquirir los datos
TriggerIndex	Numero del primer fotograma

Tabla 10: propiedades de colorMetData.

    - depthMetaData:

```
depthMetaData =

100x1 struct array with fields:

    AbsTime
    FrameNumber
    IsPositionTracked
    IsSkeletonTracked
    JointImageIndices
    JointTrackingState
    JointWorldCoordinates
    PositionImageIndices
    PositionWorldCoordinates
    RelativeFrame
    SegmentationData
    SkeletonTrackingID
    TriggerIndex
```

Código 36: Propiedades de depthMetaData en Matlab R2013a.

He aquí una breve explicación de sus campos:

Campo	Descripción
IsPositionTracked	Una matriz de 1x6 booleanos (falso, verdadero) para el seguimiento de la posición de cada uno de los seis esqueletos. 1 indica que la posición ha sido seguida y 0 que no.
IsSkeletonTracked	Una matriz de 1x6 booleanos (falso, verdadero) para el estado de un seguimiento de cada uno de los seis esqueletos. 1 indica que se hace un seguimiento y 0 que no.
JointImageIndices	Si la propiedad BodyPosture de la cámara de profundidad es *Standing*, es decir, el usuario está de pie, es una matriz de 20x2x6 de números de coma flotante (double) de coordenadas (x,y) de 20 articulaciones en píxeles con respecto a la imagen en color, para los seis posibles esqueletos. Si la propiedad BodyPosture de la cámara de profundidad es *Seated*, la matriz seria de 10x2x6 debido a las 10 articulaciones.
JointTrackingState	Una matriz de enteros de 20x6 que contiene valores enumerados (predefinidos) para la precisión de seguimiento de cada articulación de los seis esqueletos. Los valores son: 0 no seguida, 1 posición inferida, 2 posición seguida.
JointWorldCoordinates	Si la propiedad BodyPosture de la cámara de profundidad es *Standing*, es una matriz de 20x3x6 de numero de coma flotante (double) de coordenadas (x,y,z) de las 20 articulaciones en milímetros desde el sensor, para los seis esqueletos seguidos. Si la propiedad BodyPosture de la cámara de profundidad es *Seated*, la matriz seria de 10x2x6 debido a las 10 articulaciones.
PositionImageIndices	Una matriz de 2x6 de números de coma flotante de coordenadas (x,y) de cada esqueleto en píxeles respecto a imagen en color
PositionWorldCoordinates	Una matriz de 3x6 de números de coma flotante de coordenadas (x,y,z) de cada esqueleto en metros con respecto al dispositivo
SegmentationData	Una matriz que tiene la misma cantidad de píxeles que la imagen en color y el valor de cada píxel corresponde al índice de la persona en el campo de visión que es más cercano en esa posición del píxel.
SkeletonTracking	Una matriz de enteros de 1x6 (un vector) que contiene los ID de rastreo de los seis esqueletos. El ID de seguimiento se genera por Kinect y cambia de una adquisición a otra.

Tabla 11: propiedades de depthMetData de MatLab R2103a.

En este proyecto se está utilizando la versión R2013a de MatLab. Las versiones R2013b y R2014a, tienen dos campos más en *depthMetaData*:

- **JointDepthIndices**: Si la propiedad BodyPosture de la cámara de profundidad es *Standing*, es una matriz de 20x2x6 de números de coma flotante (double) de coordenadas (x,y) de 20 articulaciones en píxeles con respecto a la mapa de profundidad, para los 6 posibles esqueletos. Es decir, las posiciones de las articulaciones en el mapa de profundidad o la imagen que devuelve la cámara de profundidad. Si la propiedad BodyPosture de la cámara de profundidad es *Seated*, la matriz seria de 10x2x6 debido a las 10 articulaciones.
- **PositionDepthIndices**: Una matriz de 2x6 de números de coma flotante de coordenadas (x,y) de cada esqueleto en píxeles respecto a la imagen de profundidad.

Una vez se han grabado los 100 fotogramas, paramos las cámaras.

```
stop([colorVid depthVid]);
```

Código 37: parada de captura de fotogramas.

Configurar el seguimiento del esqueleto

Entre las propiedades de la cámara de profundidad (depthSrc) existen tres que están directamente relacionadas con el seguimiento del esqueleto: TrackingMode (sus valores son: Skeleton, Position o off), BodyPosture y SkeletonToTrack.

Para activar el seguimiento del esqueleto:

```
set(depthSrc, 'TrackingMode', 'Skeleton')
```

Código 38: activación de seguimiento del esqueleto en Matlab R2013a.

Acceder a los datos del esqueleto

Los datos del esqueleto son accesibles desde la cámara de de profundidad como parte de MetaData.

[frameDataDepth, timeDataDepth, **metaDataDepth**] = getdata(**depthVid**);

Kinect es capaz de seguir la posición de 6 personas que están en su campo de visión y puede seguir activamente los esqueletos de 2 de ellos.

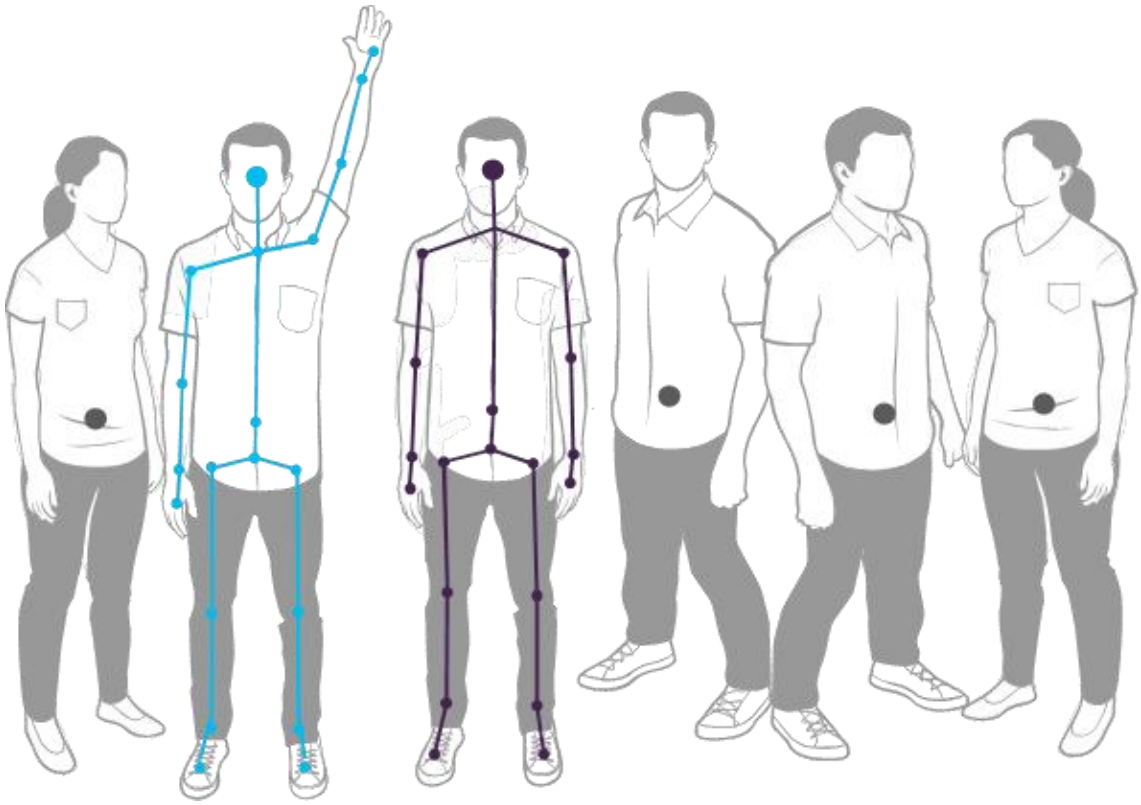


Figura 91: ejemplos de dos esqueletos y cuatro posiciones capturadas. (*)

(*) Fuente: http://passiondev.files.wordpress.com/

Vamos a recoger 100 fotogramas como hemos hecho anteriormente. Para ello cambiaremos la propiedad FramesPerTrigger de cada cámara. Después, iniciaremos el registro de datos.

```
colorVid.FramesPerTrigger = 100;
depthVid.FramesPerTrigger = 100;
start([colorVid depthVid]);
trigger([colorVid depthVid]);
```

Código 39: captura de 100 fotogramas para las dos cámaras.

A continuación, recuperaremos los datos que hemos grabado. De la cámara de color, solo nos interesan los fotogramas, no nos importan ni el tiempo ni la meta información. En cambio, de la cámara de profundidad nos interesan los fotogramas, no nos interesa el tiempo pero si nos importa y mucho la meta información. Por lo tanto

```
[frameDataColor] = getdata(colorVid);
[frameDataDepth, timeDataDepth, metaDataDepth] = getdata(depthVid);
```

Código 40: recogida de datos importantes de los dos tipos de videos.

Como hemos capturado 100 fotogramas, escogemos un fotograma al azar para trabajar con él, por ejemplo el fotograma numero 95. Para comprobar que al menos un esqueleto ha sido seguido por Kinect, lo haremos de la siguiente manera: utilizaremos estos dos campos de *depthMetaData: IsPositionTracked isSkeletonTracked*, es decir, preguntaremos si ha habido alguna posición y algún esqueleto detectados.

```
anyPositionsTracked = any(metaDataDepth(95).IsPositionTracked ~= 0)
anySkeletonsTracked = any(metaDataDepth(95).IsSkeletonTracked ~= 0)
```

Código 41: comprobación de si alguna posición o esqueleto ha sido capturado.

En caso de que Kinect haya detectado una posición de un usuario o un esqueleto, el resultado nos lo dirá.

```
anyPositionsTracked =

     1


anySkeletonsTracked =

     1
```

Código 42: confirmación de que alguna posición y esqueletos han sido capturados.

El resultado anterior nos indica que al menos un esqueleto ha sido seguido y su posición también. La propiedad isSkeletonTracked nos dirá que esqueleto de las 6 posiciones está siendo seguido.

```
metaData.IsSkeletonTracked

ans =

     1     0     0     0     0     0
```

Código 43: comprobación de que esqueleto ha sido captado.

```
trackedSkeletons = find(metaDataDepth(95).IsSkeletonTracked)
```

Código 44: metadatos del fotograma 95.

Ahora es el turno de índices. Como antes hemos indicado, la versión de MatLab R2013a contiene dos tipos diferentes de índices: JointWorldCoordinates que son las coordenadas (x,y,z) de las 20 articulaciones respecto a Kinect y JointImageIndices que son los índices (x,y) de las 20 articulaciones respecto a los fotogramas devueltos por la cámara de color.

```
jointCoordinates = metaDataDepth(95).JointWorldCoordinates(:, :, trackedSkeletons)

jointIndices = metaDataDepth(95).JointImageIndices(:, :, trackedSkeletons)
```

Código 45: índices (posición real e imagen RGB) de las articulaciones del esqueleto en el fotograma 95.

De este modo podremos ver la imagen en color y el mapa de profundidad del fotograma 95.

```
image = frameDataColor(:, :, :, 95);

prof = frameDataDepth(:, :, :, 95);
```

Código 46: fotograma RGB 95 de los 100 capturados. Fotograma de profundidad 95 de los 100 capturados.

Para ver el número de esqueleto que se han seguido. Como mínimo 0 y como máximo 2.

```
nSkeleton = length(trackedSkeletons);
```

Código 47: número de esqueleto que se ha seguido.

Si queremos ver el esqueleto superpuesto encima de imagen en color:

```
util_skeletonViewer(jointIndices, image, nSkeleton);
```

Código 48: imagen RGB (fotograma 95) con los índices de las articulaciones en la imagen RGB (jointIndices) superpuestos.

Figura 92: imagen RGB (fotograma 95) con los índices de las articulaciones en la imagen RGB (jointIndices) superpuestos.

10.8. ANEXO H: Instalación de Kinect for Windows en MatLab

10.8.1. Introducción

Antes de instalar Kinect for Windows en MatLab, debe de estar instalado Microsoft Kinect for Windows SDK, o dicho de otra manera los drivers de Kinect deben de estar instalados. Se debe contar con la versión 1.6 o mayor.

Una vez instalados los drivers, lo que vamos a hacer es instalar Kinect for Windows en el *Image Acquisition Toolbox (herramienta de Adquisición de imágenes)*. Esta toolbox servirá para:

- Interfaz entre el video, en nuestro caso Kinect y MatLab.
- Administrar propiedades de los dispositivos de harware.
- Realizar operaciones de análisis de imagen con las imágenes que está retransmitiendo el video.

Para instalar Kinect for Windows en Matlab, debemos seguir los siguientes pasos:

1. Tenemos que abrir Support Package Installler (instalador de paquetes). Se puede hacer de dos formas:
   1.1. Escribir en MatLab *supportPackageInstaller*
   1.2. Abrir el instalador de esta manera **home>Resources>Add-ons>Get Harware Support Packages**

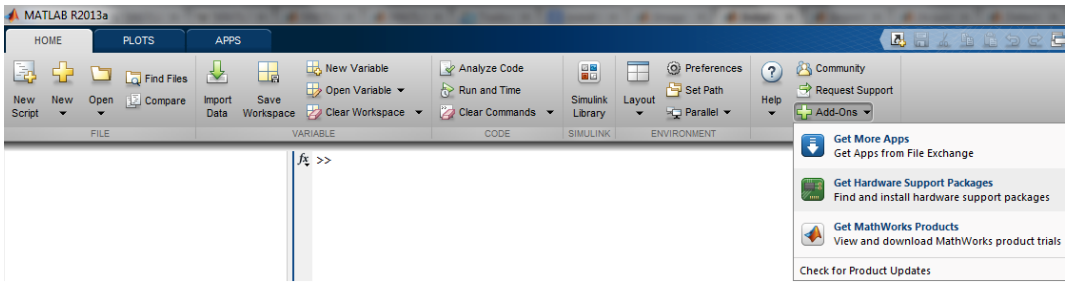


Figura 93: localización de *Get Hardware Support Packages*.

2. En la pantalla de instalar o actualizar un paquete (**install o update support package**), seleccionaremos desde Internet. El paquete será descargado e instalado desde internet.

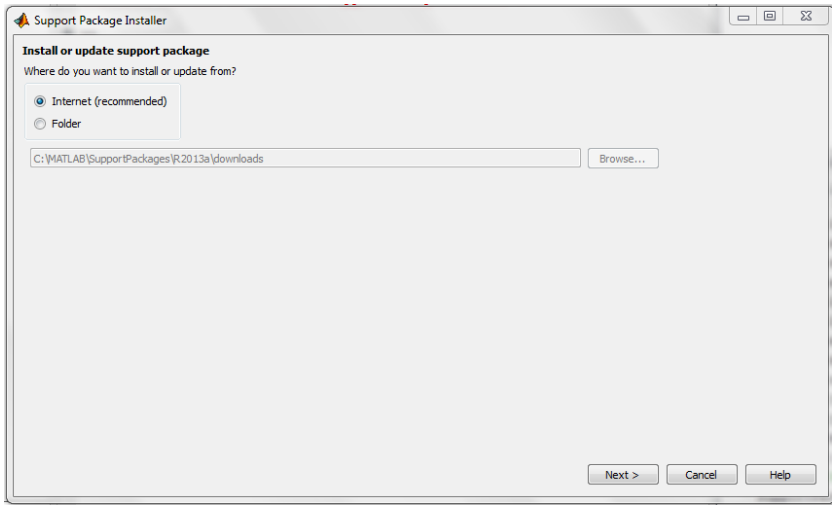


Figura 94: inicio de instalación de Kinect en Matlab.

3. A continuación seleccionaremos Kinect for Windows Runtime de la lista y seleccionamos Next.

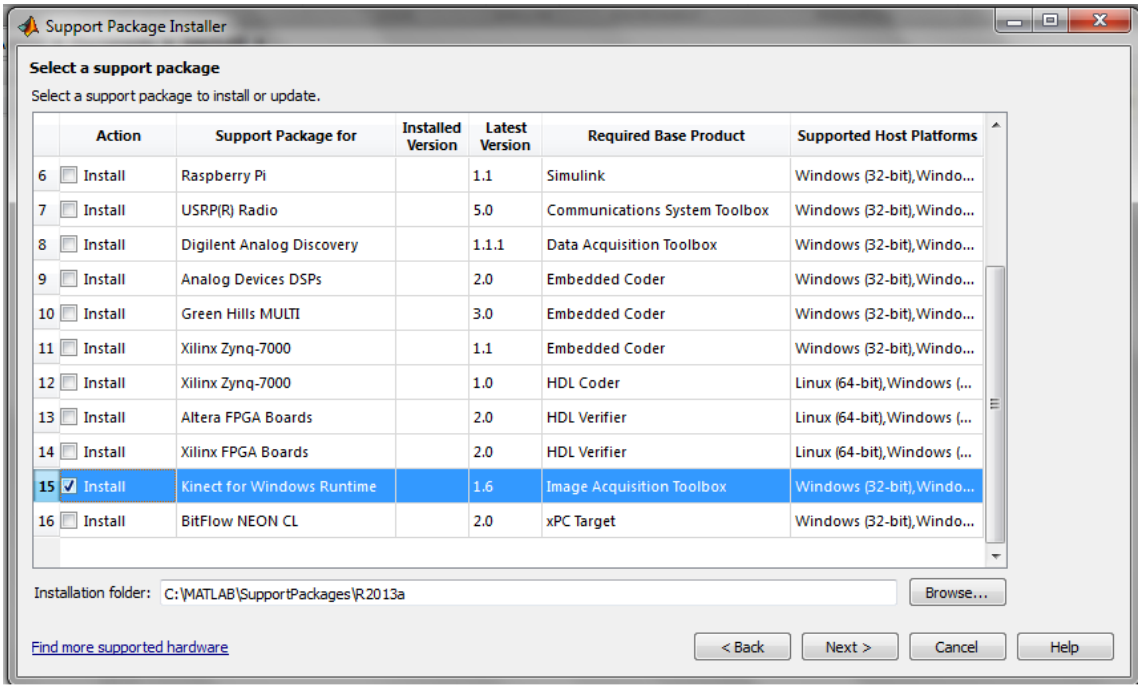


Figura 95: elección del paquete Kinect for Windows.

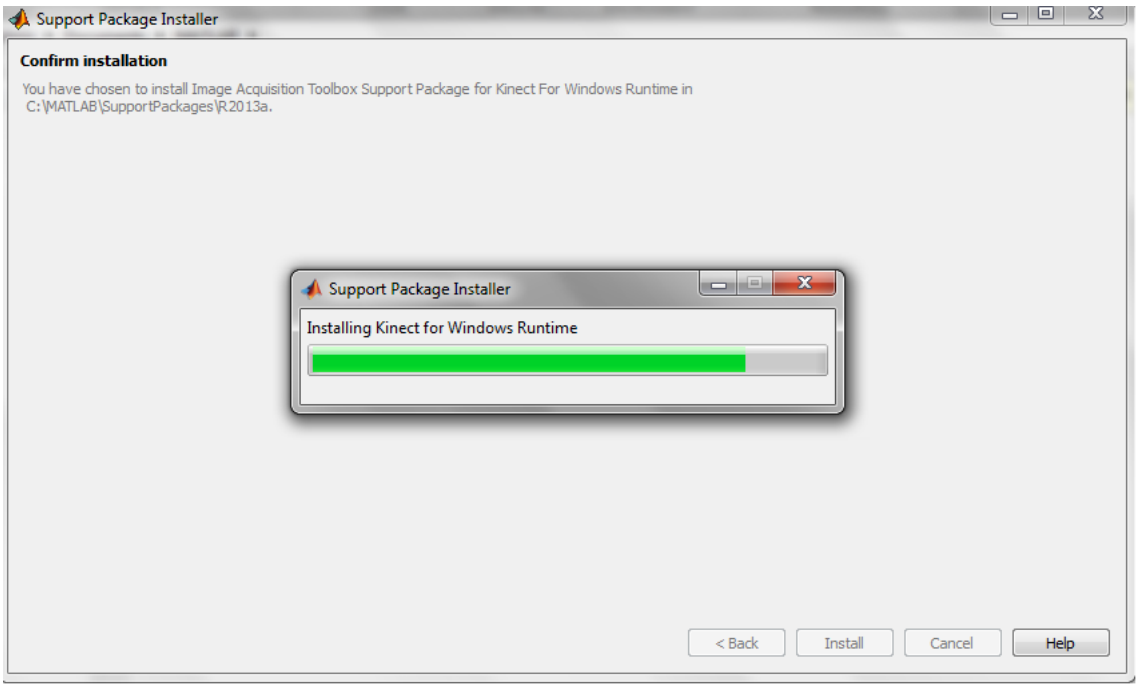


Figura 96: instalación del paquete Kinect for Windows en MatLab 1.

4. Marcamos la casilla de acuerdo con la licencia de MathWorks.

5. Se muestra la pantalla la licencia de Kinect for Windows. Revisamos la licencia y hacemos click en Next.

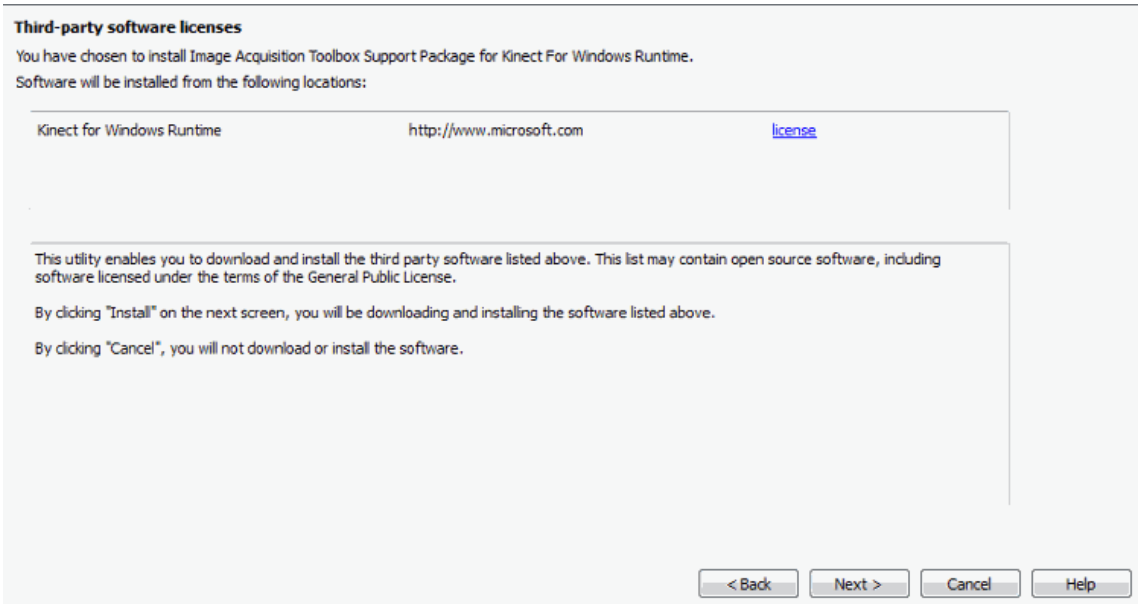


Figura 97: revisión de licencia.

6. En la pantalla de la confirmación de la instalación (Confirm Instalation), Support Package Instalation se indica el paquete que vamos a instalar y el lugar que va a ser instalado. Para confirmar la instalación hacemos cick en install.
7. Mientras vemos la barra de progresión de la instalación, nos aparecerá un dialogo de acuerdo de la licencia. Seguimos las indicaciones del acuerdo de la licencia.

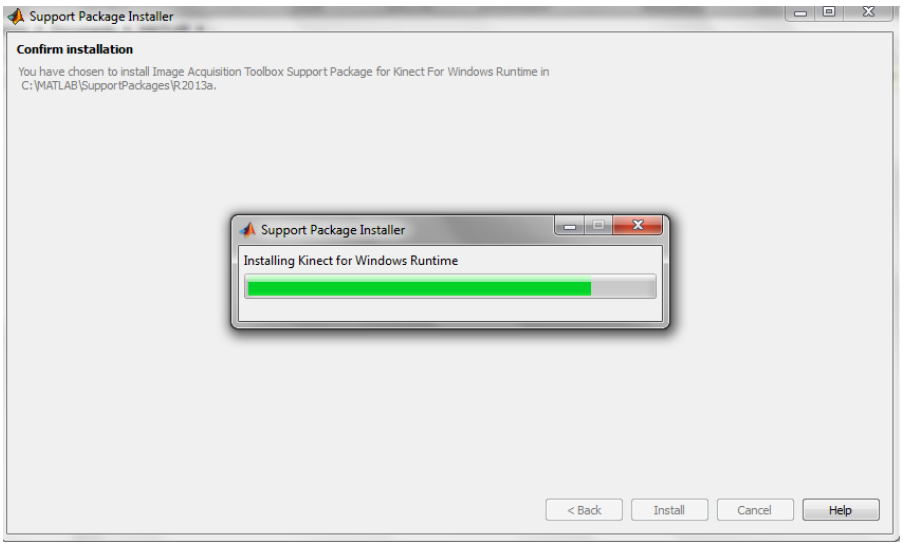


Figura 98: Instalación del paquete Kinect for Windows en MatLab 2.

8. Una vez finalizada la instalación, se mostrará un mensaje de confirmación del fin de la instalación. Hacemos click en Finish para cerrar Support Package Installer.
9. Verificamos que el paquete de kinect for Windows Runtime se encuentra en la ruta mostrada anteriormente.

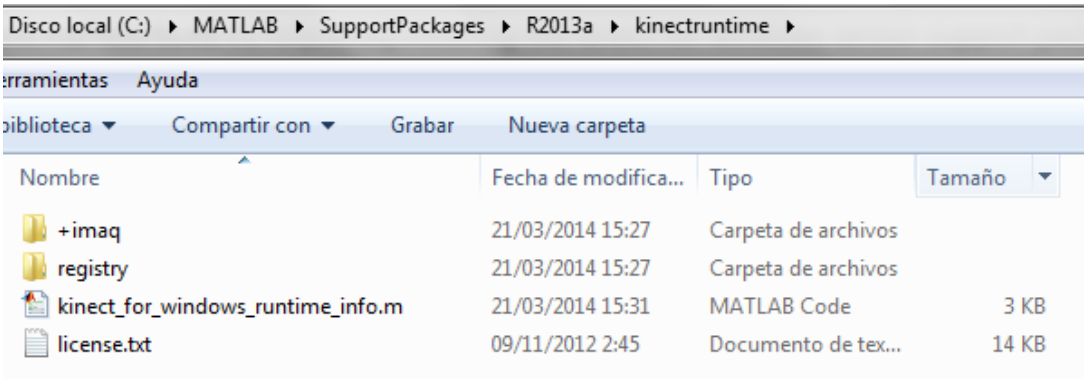


Figura 99: Confirmación de la instalacion paquete Kinect for Windows en MatLab.

Flujos de datos transmitidos por Kinect a Matlab

Secuencias de imágenes (image stream)

La secuencia de imágenes devuelve datos de imágenes en color y en otros formatos utilizando la cámara RGB. Estas son los posibles formatos en los que puede devolver.

Formato	Descripción
RawYUV_640x480	Formato Raw YUV, resolución 640x480, 15 imágenes por segundo, que es el máximo permitido
RGB_1280x960	Formato RGB, resolución 1280x960, 12 imágenes por segundo, que es el máximo permitido
RGB_640x480	Formato RGB, resolución 640x480, 30 imágenes por segundo, que es el máximo permitido
YUV_640x480	Formato YUV, resolución 640x480, 15 imágenes por segundo, que es el máximo permitido
Infrared_640x480 (*)	Formato infrarrojo, resolución 640x480, 30 imágenes por segundo, que es el máximo permitido. La imágenes en infrarrojo te permiten capturar imágenes en lugares con poca iluminación
RawBayer_1280x960	Formato Raw Bayer, resolución 1280x960, 12 imágenes por segundo, que es el máximo permitido. Este formato devuelve el patrón Bayer en bruto, así que es posible reconstruir la imagen utilizando un algoritmo
RawBayer_640x480	Formato Raw Bayer, resolución 640x480, 30 imágenes por segundo, que es el máximo permitido. Este formato devuelve el patrón Bayer en bruto, así que es posible reconstruir la imagen utilizando un algoritmo

Tabla 12: tipos de formatos capturados por la cámara de color.

(*)→En el dispositivo Kinect Xbox360, no está instalado.

Flujo del sensor de profundidad

El flujo del sensor de profundidad devuelve los datos de segmentación de las personas utilizando el sensor de profundidad de Kinect. El mapa de profundidad es una imagen en la que cada píxel es la distancia en milímetros que hay desde la cámara hasta el punto. El rango de profundidad por defecto es de 50 cm hasta 400 cm, mientras que rango de profundidad corto es de 40 cm hasta 300 cm

Estas son los posibles formatos en los que puede devolver.

Formato	Descripción
Depth_640x480	Resolución de 640x480, 30 imágenes por segundo
Depth_320x240	Resolución de 320x240, 30 imágenes por segundo
Depth_80x60	Resolución de 80x60, 30 imágenes por segundo

Tabla 13: tipos de formatos capturados por la cámara de profundidad.

Flujo de datos de esqueleto

El flujo de datos de esqueleto devuelve los datos del esqueleto utilizando el sensor de profundidad. Contiene la posición global del esqueleto y la posición 3D de 20 articulaciones. Son 2 esqueletos los que pueden ser seguidos activamente mientras que 4 pueden ser seguidos pasivamente.

10.9. ANEXO I: Kinect

10.9.1. Introducción

Kinect es un controlador juego para la consola Xbox 360. Salió al mercado en noviembre del 2010 para la consola de Microsoft y en junio de 2011 fue desarrollado para ser controlado por Windows 7. Este dispositivo fue creado para interactuar sin mandos, utilizando los movimientos del usuario.

Este dispositivo tiene:

1. Micrófonos multi-array o micrófonos de múltiples matrices. Estos 4 micrófonos colocados en fila captan los sonidos que les llegan de todas direcciones. Estos micrófonos son capaces de diferenciar los sonidos que tienen delante del resto de los sonidos. Como cada micrófono está situado en un lugar diferente, les llegara el sonido en diferentes tiempos, por lo que son capaces de calcular la fuente y la posición del sonido teniendo en cuenta ese desfase temporal que captan los micrófonos. Además, para diferenciar la voz humana, se filtra el sonido eliminando las frecuencias que no estén en el rango 80 Hz (Herzios) y 1100 Hz.

Tabla 14: localización de los cuatro micrófonos en Kinect. (*)

(*) Fuente: http://ricardoz.blogspot.com.es/

2. Cámara RGB. Esta cámara captura la imagen en color. Una imagen es un conjunto de píxeles. Cada píxel de la imagen está compuesta por un valor de RGB de 24 bits (estándar desde 2005). Una imagen RGB tiene 3 canales o componentes: rojo (Red), verde (Green) y azul (Blue). Dicho de otro modo, la imagen está compuesta de 3 imágenes, una por cada canal, donde cada imagen puede almacenar píxeles con valores entre 0 y 255. Cada píxel, al poder ser representado con 24 bits, se puede concluir que existen 2^{24} colores o lo que es lo mismo, 16.581.375 colores.
3. Dos cámaras de profundidad. Para obtener la imagen en 3 dimensiones utiliza un proyector y una cámara. El primero sirve para proyectar luz infrarroja y con el segundo se capta solo este tipo de luz, de esta manera la intensidad de la luz infrarroja que llega rebotada a la cámara, indica a la consola o al ordenador la profundidad a la que se encuentra ese píxel; de este modo, se crea el llamado mapa de profundidad.
4. Motor para controlar la inclinación. Con este motor podemos cambiar la inclinación horizontal de la kinect.
5. LED. Este LED indica el estado de la kinect. En caso de que esté apagado, significa que la kinect está apagado y si el LED está encendido significa que está encendido. Si está encendido, por defecto el color del LED es verde. Si el LED tiene color rojo, significa que hay que a la kinect no le llega suficiente energía (mirar los cables), el dispositivo no está nivelado,…

Figura 100: localización de las diferentes partes Kinect. (*)

(*) Fuente: http://blogdelchava.blogspot.com.es/

Estas son las especificaciones iniciales (por defecto) de la Kinect:

Característica	Especificación (valores)
Campo de visión vertical	43 grados
Campo de visión horizontal	57 grados
Rango de inclinación vertical (controlado por el motor)	[-27 grados, +27 grados]
Imágenes por segundo (para color y profundidad)	30 imágenes por segundo (30 fps, frames per second)
Rango de profundidad	1,2 - 3,5 metros

Tabla 15: Especificaciones técnicas de Kinect.

10.9.2. Tipos de Kinect

En la actualidad existen varios tipos de Kinects:

- Kinect para Xbox 360, pone "XBOX 360" en el frontis: este tipo de dispositivos están autorizados para desarrollo de programas y no soporta el "modo cercano". Se está utilizando este tipo de Kinect en este proyecto.
- Kinect for Windows, pone "KINECT" en el frontis: estos dispositivos están optimizados para experiencias en Windows, están autorizados para ser utilizados con Windows y soportan el "modo cercano".

Figura 101: kinect for Windows. (*)

(*) Fuente: http://officialandreascy.blogspot.com/

Modo cercano

Como hemos indicado antes el rango de profundidad estándar es de 1,2 a 3,5 metros. Esto implica, que los objetos que estén fuera de ese rango no serán detectados. El modo cercano nos permitirá detectar objetos a 0,5 metros de distancia.

- Kinect para Xbox One. Para la consola Xbox One.

Figura 102: kinect de la videoconsola XboxOne (*)

(*) Fuente: http://www.videojuegosyconsolas.com/

- Kinect for Windows 2.0, también llamado Kinect 2.0.

Figura 103: Kinect for Windows 2.0. (*)

(*) Fuente: http://www.gamezone.de/

Calibración de las cámaras de Kinect.

Como se puede apreciar en la imagen inferior las dos cámaras están distanciadas por 2,5 cm, lo que significa que van a captar dos imagines diferentes.

Figura 104: distancia entre sensor de profundidad y cámara de profundidad y distancia entre cámara de color y de profundidad. (*)

(*) Fuente: http://ricardoz.blogspot.com.es/

Existen aplicaciones que ayuda a calibrar las dos imágenes. [20]

10.9.3. Microsoft Kinect for Windows SDK

Un SDK (*Software Development Kit* o Kit de Desarrollo de Software) es un conjunto de herramientas de desarrollo de software que permite al programador crear aplicaciones para un sistema concreto, en este caso para el dispositivo Kinect. Salió al mercado en junio de 2011 y no era de uso comercial. Provee al programador herramientas para crear aplicaciones en los lenguajes de programación C++, C# o Visual Basic utilizando Microsoft Visual Studio 2010. Incluye las siguientes características.

- Corrientes de datos sin procesar: Acceder a corrientes de datos de bajo nivel del sensor de profundidad, sensor de la cámara de color y a los cuatro micrófonos multi matriz.
- Seguimiento del esqueleto: La capacidad de seguir la imagen del esqueleto de 1 o 2 personas en movimiento que estén dentro del campo de vista de Kinect.
- Capacidades avanzadas de audio: Incluyen capacidades como la supresión del ruido acústico o del eco, identifican la fuente del sonido y la integración de un API (*Application Programming Interface* o Interfaz de Programación de Aplicaciones) similar al reconocimiento de Voz de Windows. Esta aplicación se encuentra en la siguiente ubicación: *Menu inicio/programas/Accesorios/Accesibilidad*. Esta API sirve para controlar el equipo con la voz. El usuario puede decir comandos a los que el equipo responderá y también puede dictar frases al equipo. Para poder usar el Reconocimiento de Voz hay que conectar un micrófono al equipo, pero esto ya estaría hecho ya que Kinect dispone de cuatro micrófonos.
- Código de muestra y documentación.

*

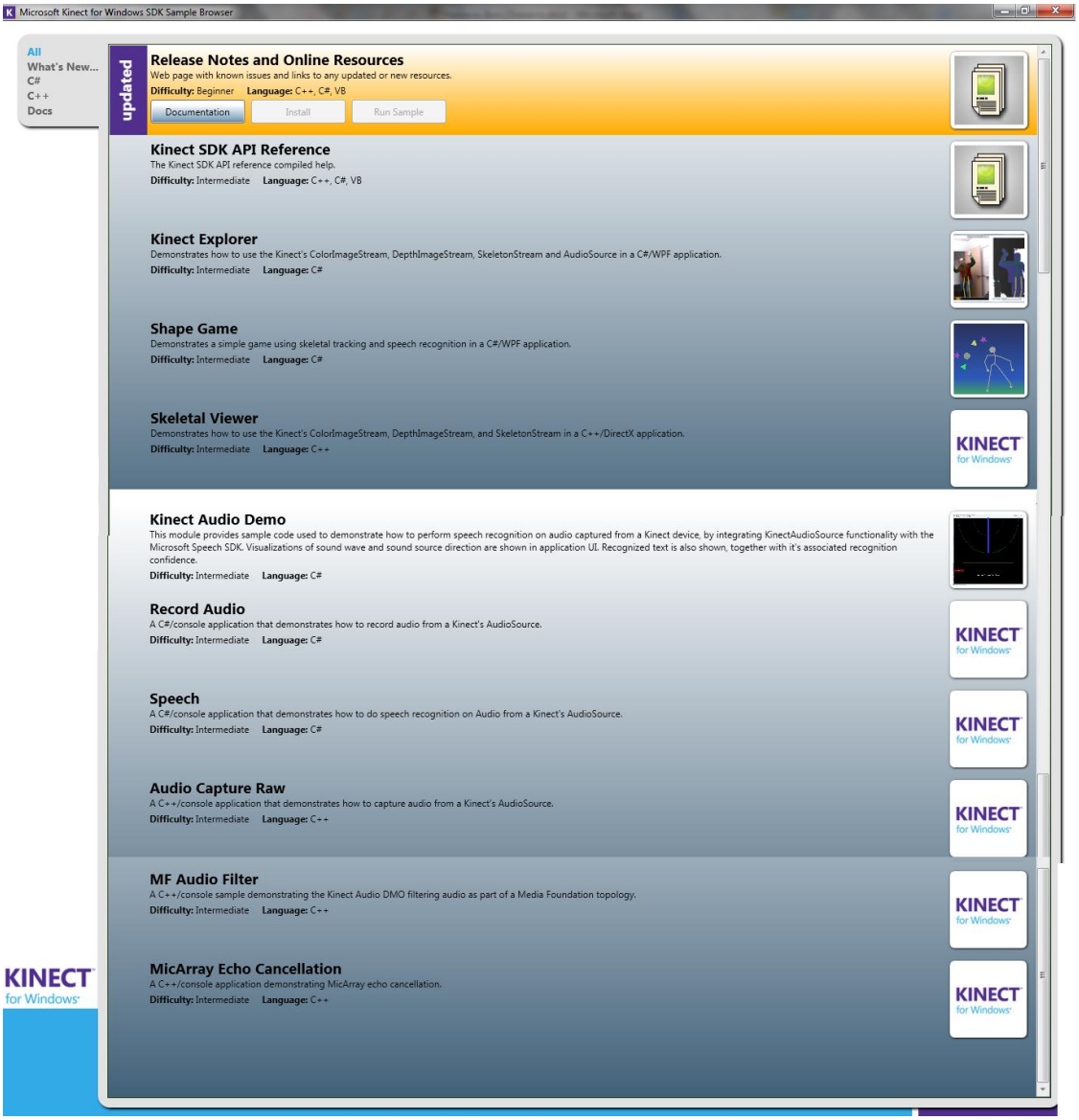


Figura 105: Microsoft Kinect for Windows SDK.

Para instalar Kinect SDK, necesitamos tener Microsoft Visual Studio. Nos descargamos desde la página oficial de Microsotf Kinect for Windows. La instalación debe de hacerse sin la Kinect enchufada al ordenador. Kinect SDK sirve tanto para arquitecturas de 32 bits como para las de 64 bits. Una vez descargado, podemos ejecutarlo inmediatamente. A continuación seguimos los pasos del asistente de instalación. Una vez instalado y con la Kinect conectada al ordenador, se puede verificar la correcta instalación. Debemos observar que el LED de la Kinect está encendido y en el administrador de dispositivos, dentro de Microsoft Kinect deben aparecer los siguientes nodos:

- Microsoft Kinect Audio Array Control.
- Microsoft Kinect Camera
- Microsoft Kinect Security Control o Microsoft Kinect Device

De esta manera podemos comprobar que los drivers de Kinect han sido instalados correctamente en nuestro ordenador.

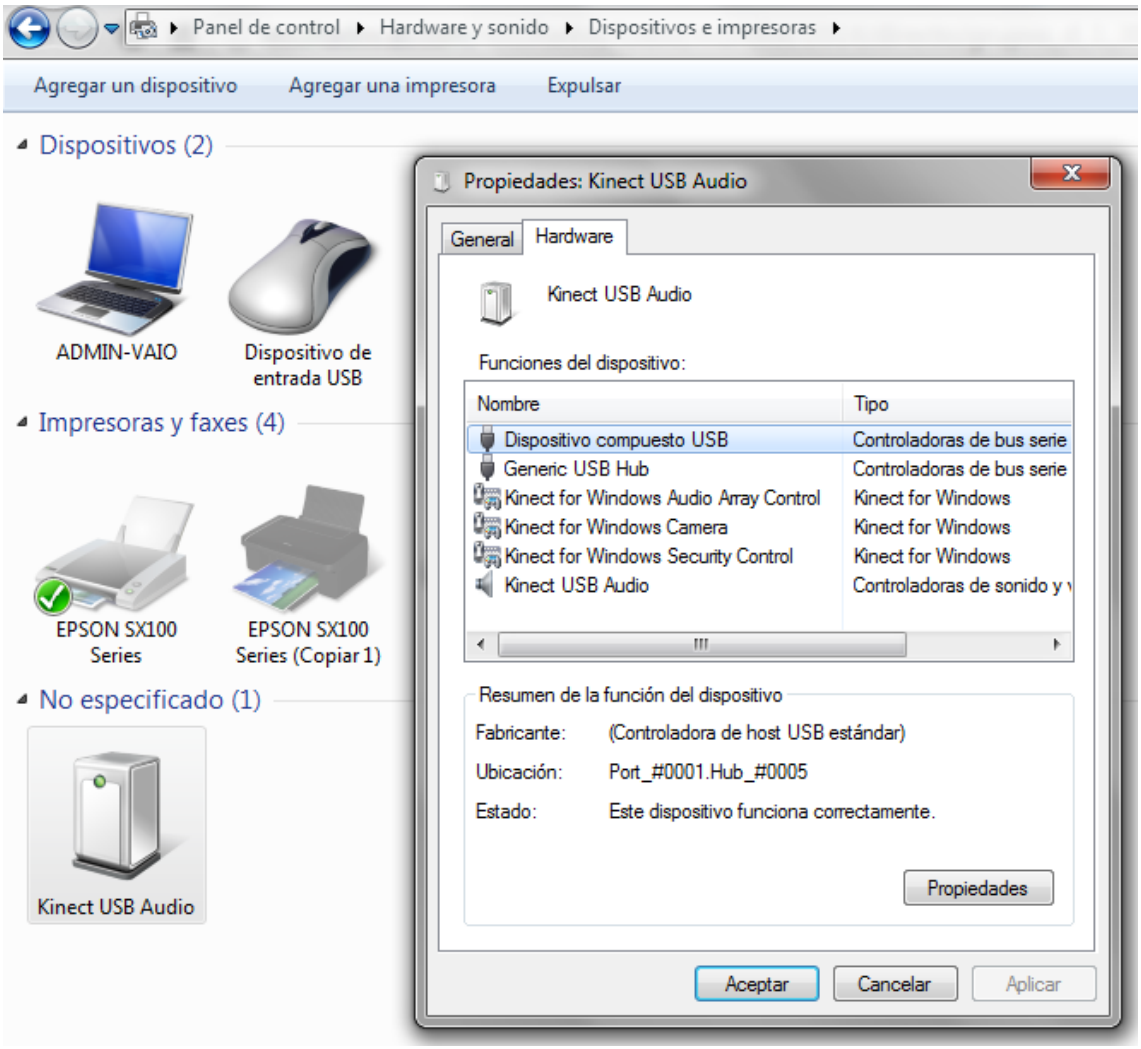


Figura 106: comprobación de la instalación de los drivers de Kinect.

Microsoft Kinect for Windows versión 1.5

En mayo de 2012 Microsoft publicó una nueva versión. Añadía nuevas características, estaba traducido en diferentes idiomas y fue presentado en 19 países.

- Incluía 'Kinect Studio' una nueva API que permitía a los desarrolladores grabar, reproducir y depurar videos de los usuarios para interactuar con las aplicaciones.
- Permitía seguir esqueletos de usuarios sentados, así como a los usuarios de pie.
- Añadía cuatro idiomas aparte del inglés para el reconocimiento de voz y también diferencia entre diferentes dialectos regionales. Inglés (Reino Unido, Irlanda, Australia, Nueva Zelanda, Canadá), francés (Francia, Canadá), italiano, Japones y Español (España, Mexico).

Microsoft Kinect for Windows versión 1.6

Esta versión fue publicada en octubre de 2012 y estas son algunas de las nuevas características que posee:

- Capacidad de detectar la orientación de dispositivo.
- Capacidad de regular aspectos de la cámara como brillo, exposición
- Soporte para Windows 7
- Soporte para Visual Studio 2012.
- Soporte para trabajar con varios dispositivos Kinect en un mismo espacio.
- Soporte para detectar un único esqueleto con varios sensores en un mismo espacio.

Microsoft Kinect for Windows versión 1.7

Esta versión fue publicada en marzo de 2013 y estas son algunas de las nuevas características que posee:

- Kinect Interactions: ofrece a los desarrolladores crear aplicaciones que utilicen los gestos más comunes de la gente.
- Kinect Fusion: ofrece la posibilidad de "renderizar" en tiempo real imágenes en 3D.

Microsoft Kinect for Windows versión 1.8

Esta versión fue publicada en septiembre de 2013 y estas son algunas de las nuevas características que posee:

- Kinect Background Removal: Tal y como dice su nombre se trata de un API que ofrece la funcionalidad de eliminar o cambiar el fondo de la imagen de un usuario.
- Servidor Web para flujos de datos de Kinect: Se trata de un componente que aporta a las aplicaciones web hechas en HTLM5 la capacidad de poder acceder a datos que la Kinect obtiene. De esta manera podemos utilizar las interacciones de la Kinect para controlar una web hecha en HTML5.

Microsoft Kinect for Windows versión 2

Está programado que saldrá a luz en verano de 2014 y está basado en la tecnología de la Kinect de la Xbox One y la Kinect 2.0. Estas son algunas de las nuevas características que posee:

- Mayor rango de visión: 70 grados en horizontal (antes 57 grados) y 60 grados en vertical (antes 43 grados)
- No tiene motor de inclinación
- Mayor resolución. 1920 x 1080 píxeles Full HD (antes 640x480). Permite detector con mas precisión. Capacidad de diferenciar la orientación del cuerpo incluyendo los manos y pudiendo incluso diferenciar los dedos.
- Mejora el rango de profundidad del sensor. Pasa a ser de 1,2 - 3,5 metros a 0,5-4,5 metros.
- USB 3.0: Aumenta la velocidad de la configuración con el ordenador, los datos fluyen más rápido y esto disminuye la latencia del sensor. Pasa de 90 ms a 60 ms.
- Captación y reconocimiento de movimientos a oscuras.

10.9.4. Código abierto

En noviembre de 2010, la empresa norteamericana Adafruit ofreció una recompensa para un controlador de código abierto para Kinect. El 10 de noviembre, Hector Martin ganó el premio, usando métodos de ingeniería inversa con Kinect y controlador para GNU/LINUX que permite el uso de la cámara RGB y las funciones de profundidad. [3] [13]

Se creó la comunidad OPENKINECT, una comunidad abierta para personas interesadas en dar uso al dispositivo Kinect. Trabajan en bibliotecas libres, de código abierto, que permite a Kinect ser utilizada en Windows, Linux y Mac. [1]

También podemos encontrar en Internet una aplicación de código abierto llamada KinectA. Esta aplicación sirve para el seguimiento del movimiento. Podemos configurarlo para que siga las manos, el esqueleto, objetos… Tiene su propia interfaz con múltiples funcionalidades. [14]

Podemos encontrar también otro driver de código abierto de Kinect llamado OPENNI. Fue creado por la empresa PrimeSense, que fue adquirida por Apple en noviembre de 2013. [15]

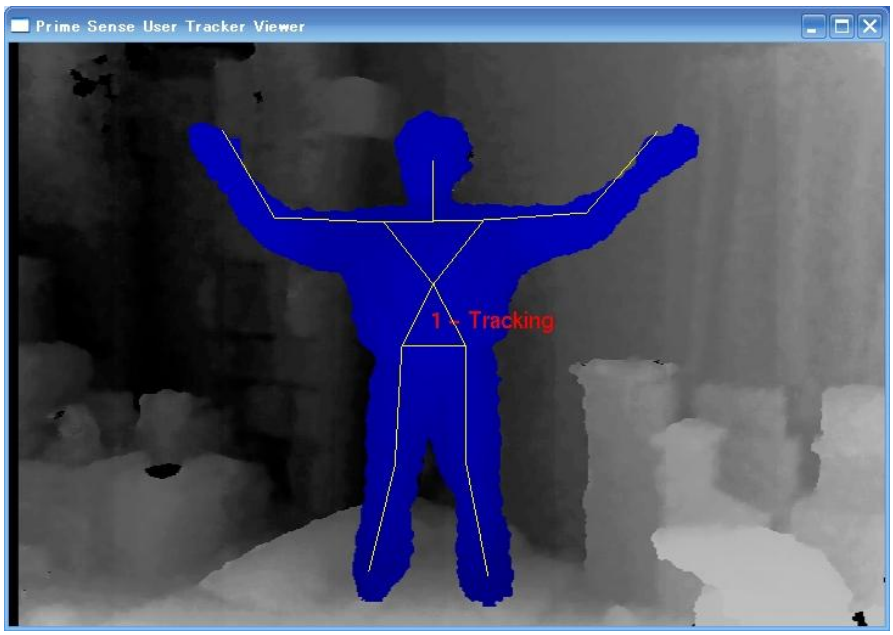


Figura 107: Seguimiento de esqueleto que realiza OPENNI. (*)

(*) Fuente: http://f.hatena.ne.jp/

11. PRESUPUESTO ORIENTATIVO

Herramientas	Coste
Kinect para Xbox 360	150,00 €
Total	150,00 €

Tabla 16: coste de kinect para Xbox 360.

Hora invertidas	Coste
343	343 x 36,49 € *
Total	12516,07 €

Tabla 17: coste en euros de horas invertidas.

* Fuente:
http://carlospologil.com/post/13196589163/35-euros-hora

12. Gestión del tiempo del proyecto

Fase	Nombre tarea	Horas estimadas	Horas reales
0	Diseño del sistema desde el punto de vista empresarial	5	6
1—makehuman	1.1) Estudiar de Python	5	7,5
	1.2) Análisis de Makehuman	25	28
	1.3) Documentar localización datos a modificar	15	12
	1.4) Reuniones	2	3
	1.5) Seguimiento y control	3	3
Horas totales		**50**	**53,5**
2—Blender	2.1) Estudio de Blender	10	21
	2.2) Crear ropa con Blender	35	45
	2.3) Documentar como crear ropa con Blender para Makehuman	20	21
	2.4) Reuniones	2	2
	2.5) Seguimiento y control	3	3
Horas totales		**70**	**92**
3—kinect	3.1) Documentar como introducir a mano las medidas en Makehuman	5	2,5
	3.2) Análisis del funcionamiento de Kinect y MatLab	35	30
	3.3) Calculo de medidas corporales obtenidas mediante Kinect	90	100
	3.4) Pruebas	10	13
	3.5) Reuniones	5	8
	3.6) Seguimiento y control	5	10
Horas totales		**150**	**163,5**
4—Base de datos	4.1) Análisis de requisitos de la aplicación web	15	12
	4.2) Estudio de creación base de datos	10	10
	4.3) Reuniones	5	3
	4.4) Seguimiento y control	5	3
Horas totales		**35**	**28**
Total proyecto		**310**	**343**

Tabla 18: comparativa de horas estimadas e invertidas.

Al finalizar el proyecto hemos superado las horas estimadas en un 9%.

En los siguientes gráficos de barras se comparan el tiempo estimado con el tiempo real.

Fase 1: Estudiar makehuman

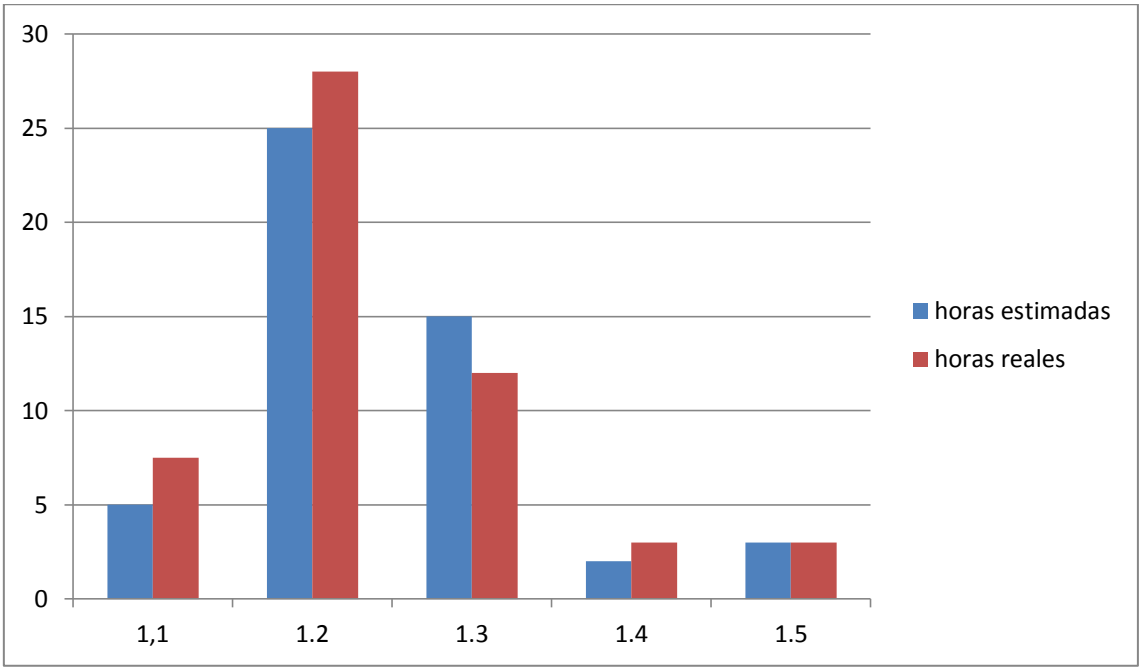


Figura 108: desviación temporal de las sub-fase de las fase 1.

Fase 2: Creación de prendas en Blender

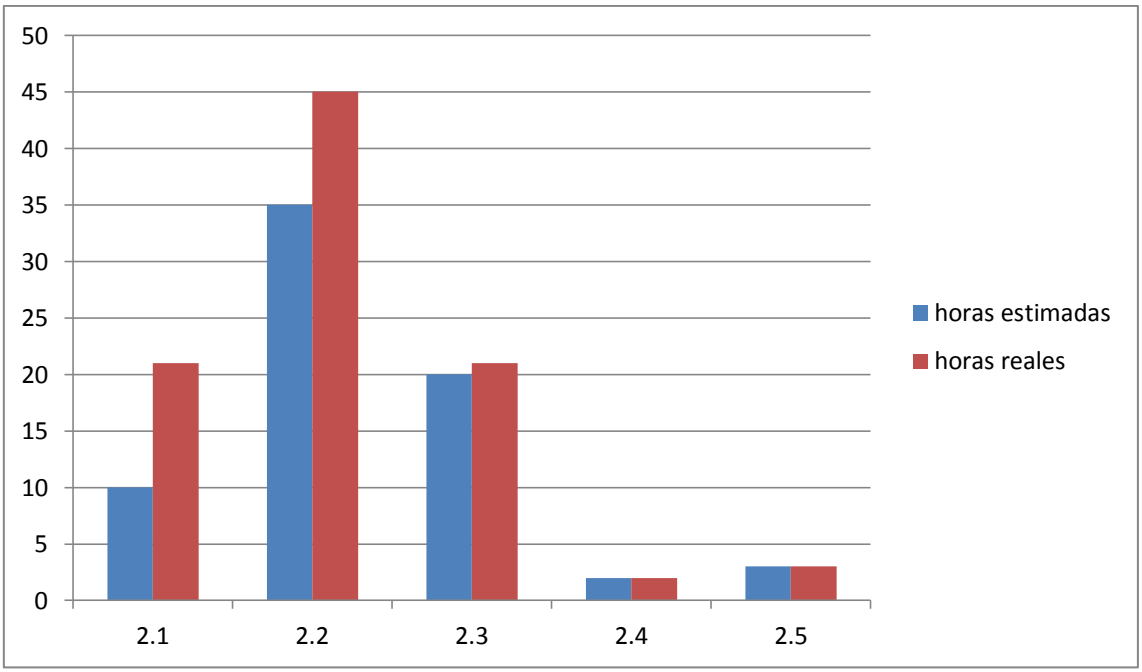


Figura 109: desviación temporal de las sub-fase de las fase 2.

Fase 3: Calcular medidas a partir de kinect

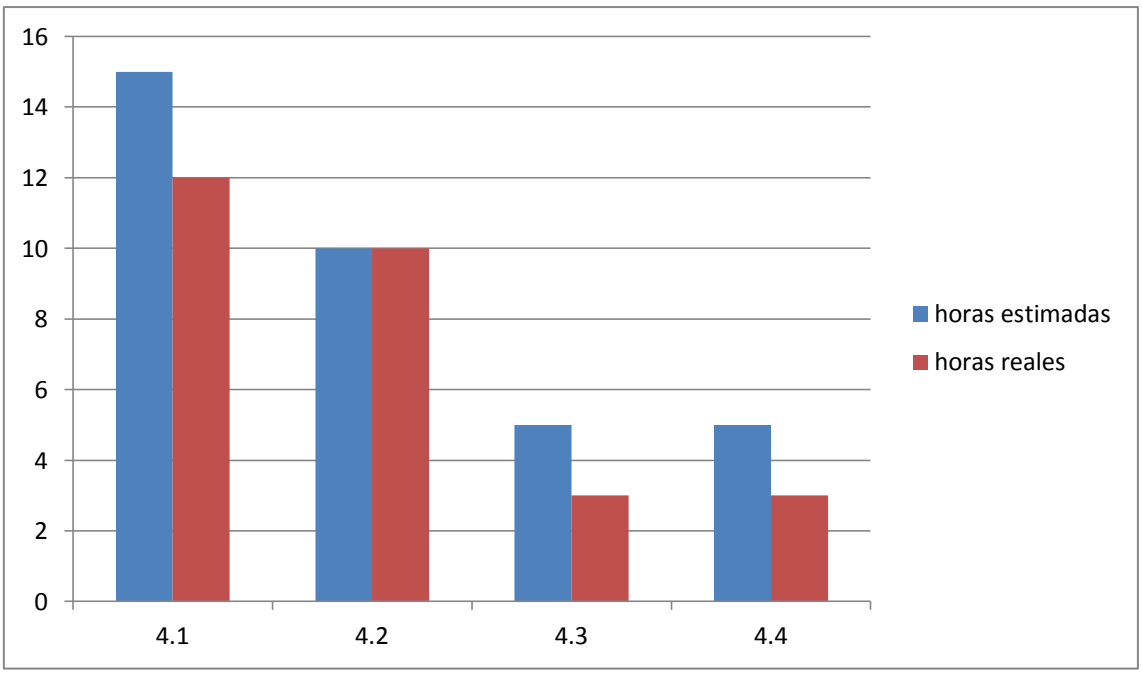


Figura 110: desviación temporal de las sub-fase de las fase 3.

Fase 4: Estudio de Base de datos de usuarios y prendas

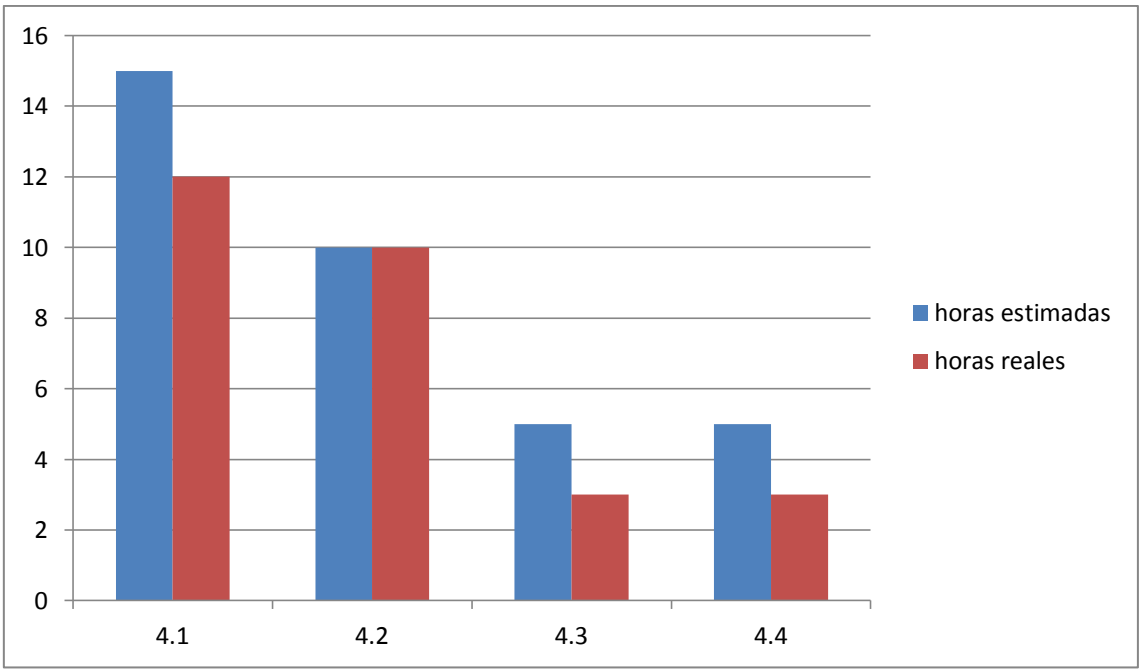


Figura 111: desviación temporal de las sub-fase de las fase 4.

13. Conclusiones y futuras aplicaciones

Este proyecto deja abiertas muchas líneas de investigación para mejorar el sistema de comercio electrónico de manera que pudieran aumentarse significativamente las ventas de las empresas que lo usen.

1. Hemos elaborado la base de datos que relaciona las medidas de los usuarios y las medidas de las prendas para que estas últimas tuvieran el tamaño idóneo para cada persona, pero sería interesante calcular las medidas del usuario mediante una simple fotografía. Esto actualmente es muy difícil, ya que aun conociendo la imagen de color (RGB) con los valores de todos los píxeles, habría que conocer la imagen de profundidad. Y claro, para ello necesitaríamos conocer la distancia del usuario respecto a la cámara y aplicar diferentes algoritmos para inferir el resto de distancias (profundidades) de los píxeles de su cuerpo respecto a la cámara.
2. Podríamos también hablar del hardware. El dispositivo Kinect es una cámara de video. Tal vez podría desarrollarse una cámara de fotos más pequeña que la Kinect, por ejemplo, una cámara estereoscópica [16], y en vez de capturar un video, podría tomar solamente las dos imágenes (color y profundidad) que necesitamos simultáneamente.
3. La calibración de Kinect está solucionada en los dispositivos Kinect 2.0 (Kinect de la consola Xbox One) y Kinect for Windows 2.0 (también conocida como Kinect for Windows v2), por lo que uno de los problemas que nos hemos encontrado en este proyecto estaría ya solucionado y las medidas corporales se podrían calcular más satisfactoriamente.
4. Adicionalmente el proyecto se podría extender a *smartphones* y tabletas ya que recientemente acaban de sacar al mercado *PrimeSense Capri*, la tecnología de Kinect aplicada a estos dispositivos.
5. Para la creación de páginas web sería interesante encontrar la forma de representar el maniquí en la web. Python es un lenguaje en constante desarrollo, y gracias a *Django*, un entorno de desarrollo web escrito en Python, se podrían generar maniquíes virtuales en la web. Esto requiere una gran inversión ya que la cantidad que datos que se necesita para crear un maniquí en tres dimensiones dificulta la carga de la web. Es conocido que cuantos menos datos tenga un web menos tiempo tarda en cargarse.

La utilización de MatLab me ha ayudado mucho para realizar un objetivo. La aplicación de las matemáticas también ha sido clave.

14. BIBLIOGRAFIA

[1] Análisis trimestre 2012 comercio electrónico en España. (Año 2012). Recuperado el 10 de diciembre de 2013, de http://onestic.com/analisis-trimestre-2012-comercio-electronico-en-espana/

{2} AENOR, Criterios generales para la elaboración de proyectos de sistema de información. UNE 157801 Madrid AENOR 2007

http://www.aenor.es/aenor/normas/normas/fichanorma.asp?codigo=N0039577&tipo=N

[3]Welcome to Blender Wiki. Recuperado el 10 de noviembre de 2013, de

http://wiki.blender.org/

[4] CHRONISTER, James. Blender Basics Classroom Tutorial Book. [on-line, PDF]. 2011. [fecha de consulta 11 de noviembre de 2013]. Disponible en: http://download.blender.org/documentation/pdf/BlenderBasics_4thEdition2011.pdf

[5] Official Makehuman. Recuperado el 10 de octubre de 2013, de

http://www.makehuman.org/

[6] Kinect for developers. Recuperado el 20 de enero de 2014, de

http://www.kinectfordevelopers.com/

[7] kinect for Windows. Recuperado el 21 de enero de 2014, de

http://www.microsoft.com/en-us/kinectforwindows/

[8] MathWorks Help. Recuperado el 10 de enero de 2014, de

http://www.mathworks.es/es/help/

15. ENLACES DE INTERÉS

[1] OPENKINECT. (Año 2010). Recuperado el 12 de diciembre de 2013, de http://openkinect.org/wiki/Main_Page

[2] Open Kinect driver(s) released (Año 2010). Recuperado el 12 de diciembre de 2013, de

http://www.adafruit.com/blog/2010/11/10/we-have-a-winner-open-kinect-drivers-released-winner-will-use-3k-for-more-hacking-plus-an-additional-2k-goes-to-the-eff/

[3] El cántabro Héctor Martin, primero en 'hackear' el sensor de juego Kinect de Microsoft. (Año 2010), Recuperado el 12 de Marzo de 2013, de

http://tecnologia.elpais.com/tecnologia/2010/11/11/actualidad/1289469664_850215.html

[4] Probador virtual para conseguir tu look ideal (año 2011), recuperado el 1 de octubre de 2013, de

http://www.theunlimitededition.com/moda/coolhunting/probador-virtual-para-conseguir-tu-look-ideal

[5] Probador virtual del Corte Inglés. [Año 2011], Recuperado el 1 de octubre de 2013, de

http://www.comocomprarporinternet.com/probador-virtual-el-corte-ingles

[6] 365 looks [Sin fecha], Recuperado el 2 de octubre de 2013, de

http://365looks.com/

[7] Zugara [sin fecha], Recuperado el 2 de octubre de 2013, de

http://zugara.com/

[8] Fitnect [sin fecha], Recuperado el 2 de octubre de 2013, de

http://www.fitnect.hu/

[9] Virtual Fitting Room. [Sin fecha], Recuperado el 1 de octubre de 2013, de

http://fits.me/

[10] WEISS, Alexander, HIRSHBERG, David, BLACK, Michael J. Home 3D body scans from noisy image and range data [en linea]. Brown University. [Fecha de consulta: 3 de diciembre de 2013]. Disponible en:

http://cs.brown.edu/people/aweiss/papers/weissICCV11.pdf

[11] SHOTTON, Jamie, FITZIBBON, Andrew, COOK, Matt, SHARP, Toby, FINOCCHIO, Mark. Real-time human pose recognition in parts from single depth images. [En linea]. Brown University. [Fecha de consulta: 3 de diciembre de 2013].Cambridge University. Disponible en:

http://research.microsoft.com/pubs/145347/bodypartrecognition.pdf

[12] LIVINGSTON, M.A., SEBASTIAN, J, AI, Z. DECKER, J. Perfomance measurements for the Microsoft Kinect Skeleton. United States Naval Research Laboratory. [En linea]. [Fecha de consulta: 3 de diciembre de 2013]. Brown University. Disponible en:

http://www.nrl.navy.mil/itd/imda/livingston-performance-measurements-microsoft-kinect-skeleton

[13] We have a winner-Open Kinect drivers(s) released. (Año 2010). Recuperado el 6 de diciembre de 2013, de

http://www.adafruit.com/blog/2010/11/10/we-have-a-winner-open-kinect-drivers-released-winner-will-use-3k-for-more-hacking-plus-an-additional-2k-goes-to-the-eff/

[14] KinectA. (Sin fecha). Recuperado el 6 de diciembre de 2013, de

http://kinecta.mihoo.de/

[15] OPENNI. (Año 2010). Recuperado el 6 de diciembre de 2013, de

https://github.com/OpenNI/OpenNI

[16] Wikipedía: cámara estereoscópica. Recuperado el 4 de abril de 2014, de

http://es.wikipedia.org/wiki/C%C3%A1mara_estereosc%C3%B3pica

[17] Wikipedia: OPENGL. Recuperado el 15 de diciembre de 2013, de

http://es.wikipedia.org/wiki/OpenGL

[18] Blender: Previous Versions. Recuperado el 10 de diciembre de 2013, de

http://www.blender.org/download/previous-versions/

[19] Foro Makehuman: Body measurements. Recuperado el 12 de octubre, de

http://forum.makehuman.org/viewtopic.php?f=6&t=10075

[20] Kinect calibration toolbox. Recuperado el 1 de febrero de 2014, de

http://www.ee.oulu.fi/~dherrera/kinect/

16. Seguimiento y control

Para este proyecto he tenido cada semana reuniones con los directores. Me han orientado y han sido de gran ayuda.

He mantenido 5 reuniones con el cliente y tuve su visto bueno.

Agradezco a Fadhi y Ali sus sugerencias.