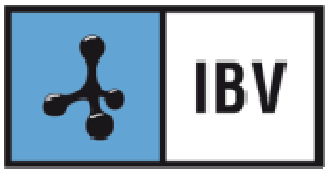
ESCUELA TECNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

UNIVERSIDAD POLITECNICA DE VALENCIA



INSTITUTO DE BIOMECÁNICA DE VALENCIA

PROYECTO FINAL DE CARRERA

DESARROLLO DE UN SISTEMA DE CAPTURA DE SILUETAS EN ANDROID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

AUTOR: Miguel Ángel Vicente Querol

DIRECTORES: Álvaro Felipe Page del Pozo

Eduardo Parrilla Bernabé

Agradecimientos

En primer lugar, me gustaría agradecer a mis tutores Álvaro Felipe Page del Pozo y Eduardo Parrilla Bernabé su paciencia y toda la ayuda y consejos que me han prestado durante la realización de este proyecto final de carrera y gracias también a toda la gente del IBV por haberse portado tan bien conmigo. En segundo lugar a todos las personas que me han acompañado, y sobre todo soportado, a lo largo de estos años tanto dentro como fuera de la carrera, gracias a todos los que habeís estado ahí. Finalmente, y de forma muy especial, a mis padres, por su comprensión y apoyo, ya que sin ellos no habría llegado hasta aquí.

A todos, gracias.

Índice

Lista de figuras

Capítulo 1: Introducción

En las siguientes líneas se hace una breve introducción al presente proyecto, exponiendo cual es su motivación y que objetivos son los que se pretenden conseguir.

1.1. Motivación del proyecto

De acuerdo a las últimas previsiones de eMarketer en el 2014 las ventas mundiales de empresa a consumidor (B2C, business-to-consumer) de comercio electrónico sobrepasaran los 1.5 trillones de dólares, siendo en el 2013 de algo más de 1.2 trillones, figura 1 [1].

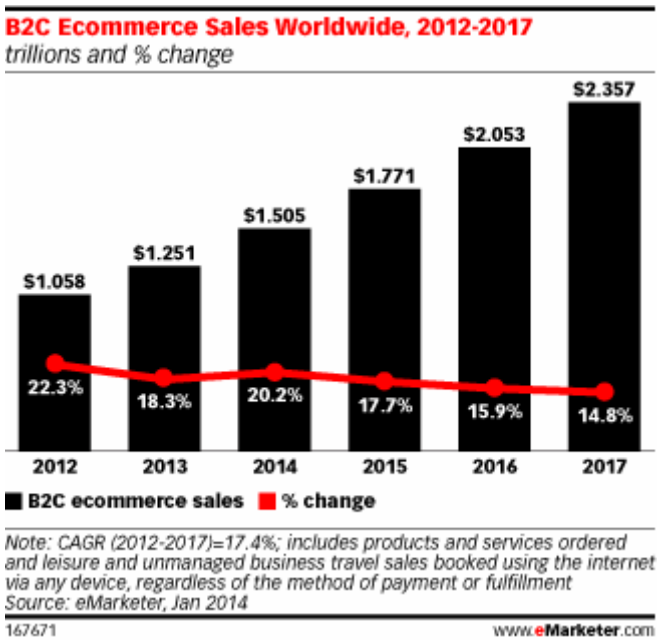


Figura 1. Ventas y previsiones de comercio electrónico. Fuente: [1]

Ystats, una de las firmas de análisis de mercados internacionales, señala en su estudio "Global Clothing B2C E-Commerce Report 2013" [2] que en el año 2013 un tercio de los internautas de todo el mundo compraron o intentaron comprar ropa online. Este informe también indica como la venta de ropa online creció en toda Europa, podemos ver por ejemplo cómo en países como Alemania esta categoría se situó como la más importante en E-Commerce o cómo en el Reino Unido casi la mitad de la población adulta navega online para comprar este tipo de productos. En Estados Unidos, en Australia y en la mayor parte de los países latinoamericanos la moda ya es la segunda categoría en cuanto a E-Commerce. En Asia esta tendencia también está al alza, por ejemplo en Japón la venta de ropa online se situa como la categoría más importante de comercio electrónico.

En España, según señala el portal Rakuten en uno de sus informes del año 2013 [3], [4], un 13% de los españoles compran más artículos online que en las tiendas convencionales y que un 27,5% compran la misma cantidad, siendo la moda una de las tres principales categorías de compra online. También, según este estudio un 50% de los españoles compran ropa online de los que más de la mitad recomiendan los artículos a amigos y familiares.

Este informe también señala que en España los dispositivos móviles (tablets y smartphones) son unos de los principales medios para este tipo de compras por parte de un 6% de la personas,

aunque queda lejos del 19% de consumidores en Tailandia y en EE.UU que utilizan este tipo de dispositivos para sus compras online.

De los anteriores datos se puede concluir que el comercio online está empujando muy fuerte y que no parece que vaya a decaer a la larga y que uno de las categorías más importantes, en algunos casos incluso por encima de la electrónica de consumo, es el sector de la moda.

Pero uno de los principales problemas que tiene la industria de la moda online es la alta tasa de devoluciones. Según señala Heikki Haldre, director ejecutivo y fundador de Fits.me [36], "Casi una de cada cuatro prendas son devueltas. El 70% de estas devoluciones es porque tiene un tamaño incorrecto". Los consumidores se basan en su experiencia previa o en la tabla de tallas que suele ofrecer la propia tienda online para seleccionar la talla, pero la experiencia previa no siempre suele ser muy fiable ya que cada empresa suele usar su propio sistema de tallaje e incluso este puede cambiar en diferentes líneas de ropa dentro de la misma marca y la tabla con medidas antropométricas que cumple cada talla tampoco es muy fiable, ya que la toma de medidas por parte del usuario en su casa está sujeta a errores significativos e incluso puede ocurrir que según la medida puedan encajar en diferentes tallas de la tabla. El resultado final se traduce en un porcentaje elevado de devoluciones y unas grandes pérdidas para el vendedor, siendo el 62% los que ofrecen a los clientes devoluciones gratuitas [5].

Para intentar solucionar este problema algunos sistemas comerciales ofrecen probadores virtuales, que mediante un avatar paramétrico que representa al usuario se realizan simulaciones para poder mostrar cómo quedaría la ropa sobre este modelo. Pero estas sofisticadas herramientas resultan inservibles para el consumidor ya que no suelen ser muy precisas y la configuración del avatar 3D que representa al usuario suele tener muy poca precisión ya que se ajusta con varias medidas tomadas por el usuario.

La tendencia después del poco éxito de estos probadores virtuales es la utilización de técnicas de tratamiento de imagen, bases de datos de cuerpos escaneados en 3D y minería de datos para poder desarrollar herramientas de reconstrucción 3D del cuerpo del usuario con unas pocas fotos que permitan obtener medidas antropométricas para la asignación de la talla y que junto con la minería de datos se pueda crear un armario virtual con preferencias o hábitos de compra.

Es en este último punto donde se enmarca el presente proyecto, en el que se va a realizar una **aplicación para teléfonos móviles en Android** que sirva para capturar siluetas, es decir, la aplicación consistirá en tomar dos fotografías, una de frente y otra de perfil de una persona que tendrá que colocarse como indicaran unas siluetas que se mostraran en la pantalla del teléfono y con técnicas de tratamiento de imagen extraer el contorno de esa persona para cada fotografía.

Esta solución implica que todos los cálculos para extraer el contorno se realizaran en el propio dispositivo, al contrario que otras aplicaciones que hay en el mercado (ej: Sunfeet). Se hace así ya que el incremento en potencia en los últimos años de los teléfonos móviles ha aumentado sobremanera, permitiendo mas carga de trabajo, lo que en principio presenta las siguientes ventajas para este proyecto:

- El usuario puede ver los resultados antes de enviar nada y poder volver a hacer las pruebas necesarias por si algo sale mal y no esperar a que se le devuelvan los resultados del servidor, con el **ahorro de tiempo** que esto supone.
- La **privacidad del usuario** es total, ya que sólo se envían los datos de los contornos obtenidos, no hay que enviar ninguna fotografía.

- **El tamaño de los datos enviados es menor** (entre 10 y 15KB), lo que supone no sólo un ahorro en ancho de banda y tiempo de subida, sino también menos gasto en la tarifa de datos contratada con el operador de telefonía móvil.
- Al procesarse parte de los datos en el Smartphone se **reduce la carga de trabajo** en el servidor, lo que supone menor tiempo de espera para el usuario.

Una vez obtenidas los contornos necesarios (de frente y de perfil), con los algoritmos y métodos desarrollados en el IBV (no desarrollados en este proyecto pero que juntos formaran parte del sistema completo, a saber, captura y calculo de siluetas, envio a un servidor, reconstrucción 3D y calculo de medidas antropométricas validas), se comprobará que los contornos obtenidos son válidos demostrándose que se puede reconstruir de forma muy exacta en 3D el cuerpo del usuario y que se pueden obtener medidas antropométricas validas para su uso en moda.

1.2. Objetivos

Los objetivos que se persiguen en este proyecto se nombran a continuación:

- Desarrollar una aplicación para captura de perfiles 2D mediante dispositivos móviles, en concreto smartphones.
- Desarrollar procesado de imagen en el mismo dispositivo para obtener los datos necesarios para la reconstrucción 3D.
- Con algoritmos desarrollados en el IBV, aplicar los datos obtenidos para obtener un modelo en 3D valido.
- A partir de ese modelo 3D, con algoritmos desarrollados en el IBV, obtener medidas antropométricas relevantes para la venta on-line.
- Validar el procedimiento mediante pruebas con una muestra.

1.3. Estructura

El documento sigue la siguiente disposición:

- En el capítulo 2 se le dará al lector una breve visión de lo que es la antropometría, de que se encarga y de cómo obtener datos antropométricos. Se hará especial hincapié en las bases de datos de modelos 3D.

- En el capítulo 3 se expondrán los materiales utilizados, tanto hardware como software.

- En el capítulo 4 se le dará al lector una breve explicación de cómo se ha desarrollado la aplicación, haciendo sobre todo hincapié en los elementos más críticos.

- En el capítulo 5 se mostraran las pruebas realizadas y los resultados obtenidos.

- En el capítulo 6 se compararan los datos antropométricos obtenidos mediante el modelo 3D reconstruido con las dos fotografías y el modelo 3D escaneado.

- En el capítulo 7 se ofrecen las conclusiones y las futuras líneas de trabajo basadas en el presente proyecto.

- En el capítulo 8 se expondrá las fuentes bibliográficas empleadas.

- En el capítulo 9 se mostrará las clases que forman la aplicación así como los ficheros más importantes divididos en diversas secciones.

Capítulo 2: Antropometría

2.1. Introducción

El término antropometría proviene del griego anthropos (hombre) y metrikos (medida) y trata del estudio cuantitativo de las características físicas del cuerpo humano. El interés por conocer las medidas y proporciones del cuerpo es muy antiguo, los egipcios ya aplicaban una fórmula fija para representar al cuerpo humano. Leonardo da Vinci con “el hombre de Vitrubio” o “Canon de las proporciones humanas”, trató de describir las proporciones del ser humano perfecto, pero que no coinciden con el ser humano actual. En el siglo XVIII ya se encuentran estudios de antropometría racial comparativa por parte de antropólogos físicos. Aunque no fue hasta 1870 con la publicación de "Antropometrie”, del matemático belga Quetlet, cuando se considera su descubrimiento y estructuración científica. Es a partir de la Segunda Guerra Mundial cuando la antropometría se consolida y desarrolla, con la necesidad de datos antropométricos en la industria, específicamente la bélica y la aeronáutica.

Las dimensiones del cuerpo humano varían de acuerdo al sexo, edad, raza, nivel socioeconómico, etc, por lo que esta ciencia dedicada a investigar, recopilar y analizar estos datos, resulta una directriz en el diseño de los objetos y espacios arquitectónicos, al ser estos contenedores o prolongaciones del cuerpo y que por lo tanto, deben estar determinados por sus dimensiones [6], [7].

Estas dimensiones son de dos tipos esenciales: estructurales y funcionales. Las estructurales son las de la cabeza, troncos y extremidades en posiciones estándar, normalmente de pie o sentado. Mientras que las funcionales o dinámicas incluyen medidas tomadas durante el movimiento realizado por el cuerpo en actividades especificas. Al conocer estos datos se conocen los espacios mínimos que el hombre necesita para desenvolverse diariamente, los cuales deben de ser considerados en el diseño de su entorno. Los estudios antropométricos resultan un importante apoyo para saber la relación de las dimensiones del hombre y las de los productos y espacios que éste necesita para realizar sus actividades de forma confortable y segura sin comprometer su seguridad [8].

Por lo tanto, disponer de datos antropométricos fiables es fundamental para diseñadores de productos y espacios de uso humano. La antropometría permite crear un entorno de trabajo adecuado permitiendo un correcto diseño de los equipos y su adecuada distribución, permitiendo configurar las características geométricas del puesto, un buen diseño del mobiliario, de las herramientas manuales, de los equipos de protección individual, etc [6].

Los datos antropométricos pueden obtenerse en diversos formatos: 1D, 2D y 3D [9]. Los datos unidimensionales consisten en estaturas, longitudes y perímetros de segmentos corporales. Permiten establecer el tamaño del cuerpo humano pero no la forma. Los datos 2D consisten en siluetas o secciones corporales, son contornos formados por curvas o puntos x, y. La antropometría 3D está formada por nubes de puntos con coordenadas x, y, z que representan la superficie del cuerpo. Un ejemplo son los escaneados 3D del cuerpo, formados por nubes de puntos que suelen contener entre 20.000 y 300.000 puntos. La adquisición, el tratamiento y análisis de los datos aumenta considerablemente en complejidad desde los datos 1D a los datos 3D.

2.2. Antropometría 1D

Históricamente, la antropometría ha sido unidimensional registrada de forma manual utilizando una serie de instrumentos como pueden ser cintas métricas, calibres y tallímetros en función de las dimensiones a medir (Figura 1). El estudio de estos instrumentos se sale de este proyecto, pero se van a nombrar los más representativos, como por ejemplo:

El **antropómetro** es una escala métrica con dos ramas, una fija y otra que se desplaza que se emplea para medir dimensiones lineales y al que se le puede acoplar reglas especiales para medir diámetros. Sirve para medir segmentos corporales, grandes diámetros y alturas [10], [11].

El **calibre o pie de rey** es un instrumento utilizado para medir dimensiones de objetos relativamente pequeños, se emplea para medir grosores, espesores y distancias entre puntos. Mediante piezas que se le pueden acoplar permite medir dimensiones internas y profundidades [6].

El **compás de pliegues cutáneos** (plicómetro) se emplea para medir panículo adiposo [10].

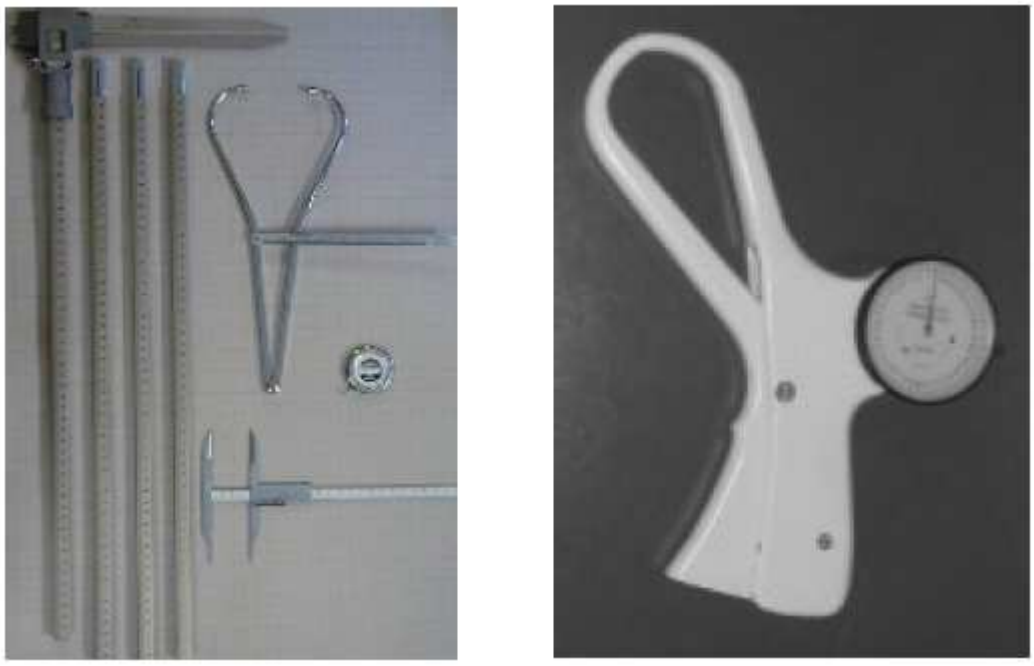


Figura 2. Equipos tradicionales de toma de datos antropométricos. Izquierda: Calibres y antropómetros. Derecha: Plicómetro. Fuente: IBV

La **cinta antropométrica** se utiliza para medir perímetros y para localización del punto medio entre dos puntos anatómicos [11].

Normalmente la lectura de estos instrumentos se hace sobre una escala que llevan grabada a lo largo de la pieza fija, aunque algunos ya incluyen una pantalla de cristal líquido que permite la lectura digital.

Los **goniómetros** y **flexómetros** se usan para medir los ángulos que forman las articulaciones [6].

En principio, la medida directa es la que proporciona los valores más precisos de las dimensiones consideradas. Como ventajas cabe destacar que el equipo es ligero, preciso y fácil

de transportar y de manejar y tiene un costo razonable. Sin embargo, tiene ciertos inconvenientes, como que el proceso de la medida y su posterior registro es laborioso y requiere experiencia y cuidado, siendo, por ello, algo lento. La calibración del material es otro inconveniente, por ejemplo las ramas del plicómetro van perdiendo fuerza su uso y la lectura de los datos será errónea.

El error del observador es la fuente principal de error en la antropometría tradicional. Incluye imprecisión en la localización de landmarks, postura del sujeto y uso del instrumental. Este error puede acentuarse con el uso de múltiples operarios, aunque hayan sido entrenados por el mismo experto y trabajen en cooperación [12].

2.3. Antropometría 2D

Consiste en el registro y análisis de contornos y secciones corporales. Los datos antropométricos en 2D suelen proceder de imágenes de las cuales se obtiene la silueta o contorno (Figura 2) o de modelos 3D sobre el cual se obtiene secciones para simplificar el análisis posterior (Figura 3).

Figura 3. Registro del contorno de la huella plantar. Izquierda: Piroscopio. Derecha Extracción del contorno. Fuente: IBV

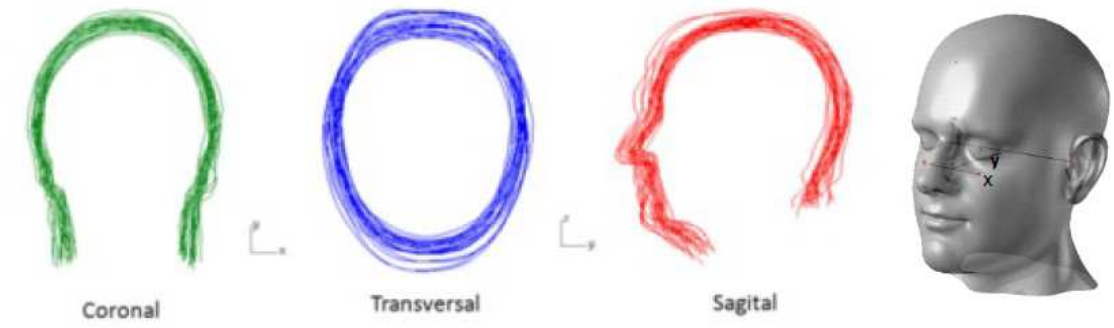


Figura 4. Secciones de una base de datos de cabezas 3D alineadas. Fuente: IBV

A diferencia de la antropometría unidimensional tradicional, la antropometría 3D requiere un procesado y análisis más complejo que incluye los siguientes pasos:

Alineado de las secciones 2D: el procedimiento de alineación puede basarse en puntos anatómicos que se superponen arrastrando la posición de la sección en 2D o puede basarse en ejes y planos definidos por puntos anatómicos concretos o por la propia forma de la sección como los ejes principales de inercia.

Análisis 2D: Una vez alineadas las secciones, el cálculo de secciones medias y percentiles es bastante complejo. Se han analizado múltiples aproximaciones, desde el análisis de Fourier hasta el análisis funcional de curvas, ajustando funciones sobre las secciones. Sin embargo, el tipo de análisis más extendido se basa en discretizar las curvas en puntos homólogos y analizar la variabilidad de las coordenadas x e y de estos puntos mediante análisis de componentes principales (PCA) o los métodos de deformación mediante matrices biortogonales [9] (Figura 5).

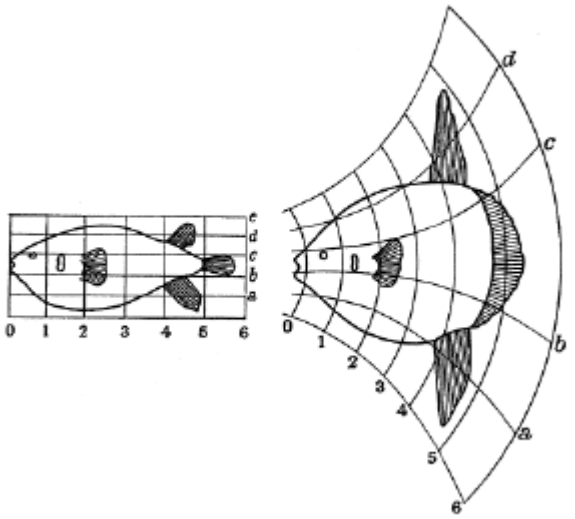


Figura 5. Estudio de la forma para caracterización de especies mediante técnicas de deformación de matrices. Fuente: IBV

Una de las dificultades principales del análisis 2D consiste en establecer el criterio de alineado, pues la variabilidad de la forma de la secciones de un grupo de sujetos depende del criterio de superposición. Además, el criterio de alineación puede definirse para alinear formas corporales entre sí que permita analizar la variabilidad de la población o puede establecerse respecto al producto a diseñar, con lo que la variabilidad de las formas corporales, serán relativas a la forma de llevar el producto.

2.4. Antropometría 3D

La antropometría digital 3D surgió con la idea de reducir el tipo de adquisición por sujeto, ya que el escaneado se reduce a pocos segundos y el software de procesado puede proporcionar las dimensiones antropométricas de forma automática, pudiendo obtenerse datos en cualquier momento que sean necesarios, ya que se puede almacenar la silueta obtenida con el escáner.

Para la realización de bases de datos y estudios antropométricos se utilizan los escáneres 3D, que pueden ser de cuerpo completo o de alguna parte en concreto del cuerpo, como los píes o la cabeza. Los hay de diversos tipos, como los de tecnología de luz estructurada como el fabricado

por la empresa francesa Telmat Industrie [13] o el desarrollado por la empresa Textile Clothing and Technology Corporation([TC]^2)[14]. Como alternativa surgen los escáneres de proyección laser, que aunque son más caros, la precisión de la forma 3D resultante es mucho mayor y se utilizan en la mayoría de estudios antropométricos nacionales. Como ejemplo de estos escáneres tenemos los de Cyberware (USA) [15] y Human Solutions (Alemanía) [16].

Un escáner 3D está formado por varias cámaras que reúnen información sobre la geometría del sujeto de estudio. Se crea una nube de puntos (cientos de miles) a la que hay que aplicar software de reconstrucción y rellenado de agujeros, que varía en función del escáner, para poder obtener la imagen en 3D de una persona. Una vez obtenía esta imagen ya se pueden extraer de forma más o menos automática las medidas antropométricas que sean necesarias [18].

Sin embargo al comparar la medidas 1D del modelo 3D con las medidas 1D del mismo sujeto tomadas de forma manual, se observan importante diferencias, lo que se lleva a considerar las medidas 1D obtenidas mediante escáneres 3D como una categoría separada de las medidas antropométricas tradicionales [18]. Otro problema que tienen los escáneres 3D es que presentan zonas de oclusión o de sombra en la zona de las axilas o la entrepierna.

Uno de los campos en el que más se utilizan este tipo de datos es en el diseño de indumentaria, aplicando técnicas morfométricas y biomecánicas para estudiar el cuerpo humano, obtener una correcta imagen 3D y poder estudiar el ajuste de prendas de ropa.

Desde la aparición de los escáneres 3D se han realizado multitud de estudios antropométricos. El primero de ellos fue el proyecto CAESAR (Civilian American and European Surface Anthropometry Resource) [19], realizado entre el año 1998 y el año 2000, fue un estudio de la población civil llevado a cabo por tres países miembros de la OTAN: Estados Unidos, Holanda e Italia. A parte de ser el primer estudio antropométrico en usar escáneres 3D, también fue el primero que realizo la OTAN sobre personal civil.

El proyecto Size-UK (2001-2002) [20] fue llevado a cabo por el sector textil para actualizar sus bases de datos de tallaje en base a las medidas antropométricas de la población. Fue una colaboración entre el Gobierno del Reino Unido con los fabricantes de moda más importantes del país y centros académicos especializados en esta línea.

También se han realizado muchos más estudios antropométricos con escáner 3D en diversos países como USA (Size USA), Francia (Estudio antropométrico de la población francesa), Alemania (Size Germany), España (Estudio antropométrico de la población femenina en España) entre otros, pero todos siguen las siguientes pautas para llevar a cabo el estudio:

1. **Diseño de experimentos**: Indicar a quien va dirigido el estudio indicando edad, género, raza, etc.
2. **Equipo de adquisición**: Tipo y número de escáneres usados.
3. **Posturas de medida**: De pie, sentado, sentado con los brazos levantados, etc.
4. **Marcadores**: Tipo y numero de marcados usados sobre el cuerpo.
5. **Ropa de escaneado**: Indicando el tipo de ropa a usar: pantalones cortos, largos, ropa ajustada, etc.
6. **Encuesta socio-demográfica**: Diversas preguntas al usuario como edad, altura, peso, etc.
7. **Extracción de medidas**: Obtención de medidas a posteriori y la forma de obtención (automática o semi-automática).

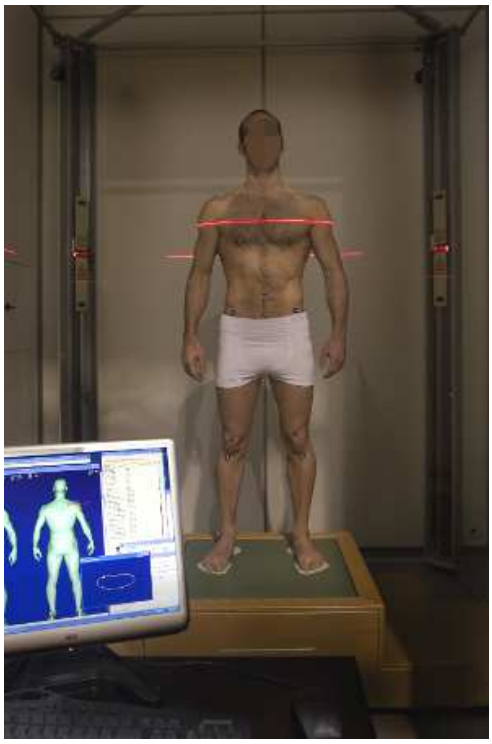


Figura 6. Sesión de escaneo 3D. Fuente: IBV

Aunque se han obtenido muchos datos de estos estudios y se han construido unas bases de datos con muchos detalles, al final estas bases no se acaban de utilizar del todo. Siempre cuesta cambiar de una forma de hacer las cosas a otra. Pero esta tendencia está cambiando, ya que están surgiendo diversos estudios que quieren aprovechar estas bases de datos sobre todo para aplicarlas al comercio on-line como puede ser por ejemplo Whatfitsme [21], SizeWand [22], Mipso [23], Truefit [24], Zafu [25], Me-ality [26] ,UPcload [27], Poikos [28], Fitsme [29], Styku [30], Bodypal [31].

2.5. Aplicación al diseño de productos

Los datos antropométricos se expresan generalmente en forma de percentiles. Un percentil expresa el porcentaje de individuos de una población dada con una dimensión corporal igual o menor a un determinado valor. El percentil es una medida de posición, es decir, que son valores que comprenden a un porcentaje determinado del conjunto de la distribución. Así, por ejemplo, el percentil 25 o P25 hace referencia al 25% de los individuos de la población considerada que tiene para la variable que se considere un valor igual o inferior al P25 de esa variable [6].

El concepto de percentil es muy útil ya que nos permite simplificar cuando hablamos del porcentaje de personas que vamos a tener en cuenta para el diseño. Por ejemplo, cuando nos referimos a la talla y hablamos del P5, éste corresponde a un individuo de talla pequeña y quiere decir que sólo un 5% de la población tienen esa talla o menos. Si nos referimos al P50, lo que decimos es que por debajo de ese valor se encuentra la mitad de la población, mientras que cuando hablamos del P95, se está diciendo que por debajo de este punto está situado el 95% de la población, es decir, casi toda la población [6].

Sin embargo el uso de percentiles de medidas antropométricas y su combinación tiene sus limitaciones, ya que estos no son aditivos y no existen individuos que tengan el mismo percentil

en todas sus dimensiones, dando como resultado el diseño de una combinación de percentiles un individuo que no existe [32].

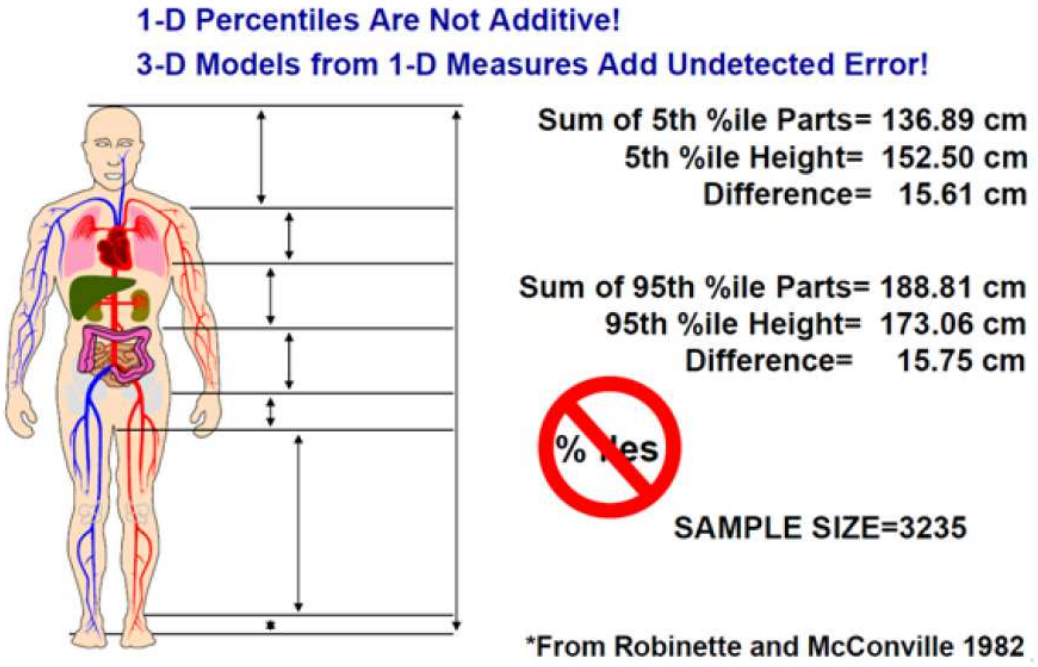


Figura 7. Propiedad no aditiva del uso de percentiles. Fuente: IBV

Estos datos son los usados por diseñadores y profesionales de la ergonomía que seleccionan los estadísticos en función del criterio usado para diseñar sus productos. Estos datos se obtienen de diferentes fuentes, como **bases de datos**, por ejemplo Humanscale (1983) [33] y Bodyspace (1986) [7] y **normas**, tanto para la confección de ropa (ISO 365:1981) o seguridad industrial (ISO 14738:2002). También se pueden usar modelos digitales 3D como los **modelos digital humanos** (DHM), que representan datos antropométricos 1D como la media, el percentil 5, el 95, etc y se suelen utilizar principalmente en el sector de la automoción y transporte, como por ejemplo Ramsis [34] de la empresa alemana Human Solutions o los **avatares,** que consisten en modelos CAD 3D que pueden ajustarse a medidas antropométricas específicas. La diferencia entre estos dos modelos 3D es que lo avatares tienen mejor acabado estético y se utilizan sobre todo en la industria de la moda con aplicaciones relacionadas con probadores virtuales y videojuegos.

Las distintas medidas antropométricas varían de una población a otra. Entre los parámetros más influyentes se puede destacar el sexo, la raza, la edad y la alimentación, por lo que hay que tener especial cuidado al elegir una fuente de datos u otra para el diseño de los productos [6].

Pero como tener una base de datos de la población objetivo es complicado, lo normal es trabajar con datos antropométricos ya publicados. Se suele usar una aproximación para ajustar el rango de parámetros antropométricos para la población objetivo [35].

La aproximación más básica es '**Procrustes**', en la que usuario es el que se ajusta al producto. Este término proviene de la mitología griega. Más extendido esta la aproximación 'Ego diseño', en la que el propio diseñador usa su cuerpo como referencia en un primer punto de referencia.

Diseñar para la media suele ser un gran error ya que puede excluir a un gran porcentaje de la población. Por ejemplo, si se diseña una entrada para la altura media, los que superen esa media no podrán pasar erguidos y tendrán que agacharse. Se suele usar cuando precisión de la dimensión no tiene importancia o su frecuencia de uso es muy baja.

El diseño para los extremos (más grande o más pequeño) suele resultar en casos como el anterior, pero, por ejemplo, en el caso de querer colocar algún panel de control se deberá tener en cuenta a las personas más bajas. Para este tipo de situaciones se suele proponer un **diseño ajustable,** que significa que el producto se puede ajustar a un gran rango de usuarios.

Pero diseño más extendido es el **diseño para la mayoría,** en el que se ajusta el diseño para un grupo razonable de la población que suele incluirse un rango que va desde el P5 hasta el P95. La aproximación ideal sería el **diseño para todos**, en el que el producto puede ajustarse a todo el conjunto de la población. Es técnicamente posible ya que la población no es infinitamente variable en cualquier dimensión.

Una vez establecido el grupo de población objetivo y su información antropométrica de referencia, y debido a que esta última información en algunos casos no es directa, se tiene que hacer una transformación de las dimensiones antropométricas en dimensiones del producto o entorno. Se debe a diversos factores entre los que cabe destacar los dos siguientes:

Por lo general los datos antropométricos son representaciones simplificadas de las variables físicas humanas, como por ejemplo los pies o las manos que cambian su antropometría y que tiene que tenerse en cuenta a la hora del diseño de guantes y zapatos para ajustarse al movimiento.

El efecto de los materiales tiene que tenerse en cuenta, no se usa el mismo material a la hora de hacer una prótesis en que el material a de ser más rígido que en el de por ejemplo una camiseta que puede ser más holgada.

Por eso también hay que hacer pruebas en usuarios y obtener información subjetiva, y no fijarse sólo en la información antropométrica, como por ejemplo medidas entre la interacción mecánica entre las superficies del cuerpo y el producto para poder ajustar mejor medidas y materiales.

El diseño ergonómico se puede resumir en la siguiente gráfica, en la que el segundo nivel puede eliminarse en función del tipo de producto.

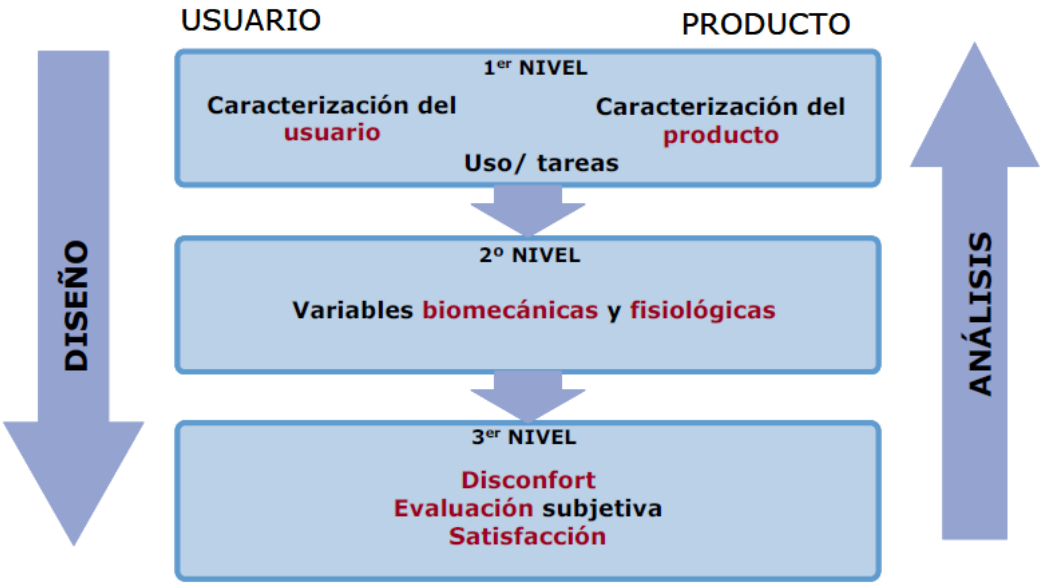


Figura 8. Aproximación metodológica para la generación de criterios de ajuste ergonómico de producto. Fuente: IBV

2.6. Comercio on-line de ropa

Uno de los principales motivos del alto número de devoluciones en la compra on-line se debe a una selección errónea de la talla. Como dice Heikki Haldre, director ejecutivo y fundador de Fits.me con sede en Londres, "Casi una de cada cuatro prendas son devueltas. El 70% de estas devoluciones es porque tiene un tamaño incorrecto" [36]. El problema se agrava si tenemos en cuenta que cada marca usa su propio sistema de tallas y no siguen un estándar. Esto supone una barrera para muchos clientes que podría traducirse en más beneficios.

El primer intento de solucionar el problema de la selección de talla de ropa en la venta online fueron los probadores virtuales utilizados en la industria de la moda pero desarrollados específicamente para venta online. Como ejemplo tenemos My Virtual Model [37] y una solución japonesa, Haoreba [38]. Presentan una forma sencilla de vestir avatares 3D personalizados en los que puede cambiarse la apariencia para asemejarse al cliente.

Figura 9. De izquierda a derecha: probador virtual de MyVirtualModel, ejemplos de este sistema y probador virtual de Haoreba de Digital Fashion. Fuente IBV

La principal desventaja es que no son representaciones reales de la prenda sobre el modelo humano por lo que la variación del ajuste cuando se modifica el modelo digital humano no es real. Estos modelos suelen primar la estética sobre el resultado. La ventaja que tienen es que es

muy fácil generar una aplicación web, tanto en la generación del modelo corporal como la generación de colecciones virtuales de vestidos.

Las nuevas propuestas para abordar el problema de selección de talla se centran en cómo realizar y gestionar una caracterización antropométrica del consumidor. Por ejemplo Poikos [28] y Upcload [27] obtienen medidas corporales utilizando una base de datos 3D a partir de imágenes de frente y de perfil del usuario capturadas mediante webcam o la cámara de un móvil. Me-ality [26] por ejemplo propone escanear el cuerpo completo del cliente en puntos habilitados. Desde el IBV, con su aplicación Sunfeet [39] para teléfonos móviles propone una reconstrucción 3D completa de los pies a partir de fotos tomadas con el móvil para la creación de plantillas personificadas.

Todas estas propuestas están intentando dar uso a bases de datos antropométricas en 3D, que están desaprovechadas, con aplicaciones que a partir de un determinado número de fotos de distintos perfiles usando ya sea cámaras web, Kinect o un teléfono móvil se puedan obtener medidas antropométricas pero realizando todo el tratamiento y posterior reconstrucción 3D en un servidor dedicado.

Para este proyecto lo que se va a realizar es todo el proceso del tratamiento de imagen en el mismo dispositivo, enviándose al final de la aplicación sólo los datos relacionados con los contornos obtenidos en la segmentación en un fichero de poco tamaño (entre 10 y 15 KB), garantizándose así la **privacidad del usuario** ya que no se tiene que enviar ninguna imagen del mismo, sólo el contorno y pudiéndose ver el **resultado de la segmentación** en el propio dispositivo, lo que supone poder volver a realizar las fotos en el caso de que el contorno no sea el correcto, evitando así tener que esperar a los resultados finales para saber si se habían hecho bien las fotos necesarias.

Una vez esté en el servidor el fichero con los contornos y los demás datos necesarios se procederá a emplear el software (desarrollado ya en el IBV, e independiente de este proyecto) que se encargará de reconstruir el modelo 3D usando las bases de datos 3D disponibles en el IBV y a partir de este modelo reconstruido se podrán obtener datos antropométricos que serán de utilidad tanto para la confección de ropa nueva como para poder, a partir de algún futuro probador virtual con un modelo 3D lo más semejante posible al cliente (o este mismo modelo 3D calculado) y modelos que representen la forma de los diferentes tejidos, probarse ropa online de forma más exacta teniendo en cuenta el comportamiento de la ropa sobre el cuerpo y con las medidas antropométricas calculadas poder ajustar mejor la talla de la ropa y así evitar las devoluciones que al final suponen perdidas para las tiendas.

Capítulo 3: Material y Métodos

El Smartphone es el medio elegido tanto para la captura de imágenes como para su posterior tratamiento para obtener los datos necesarios para la reconstrucción 3D y las medidas antropométricas necesarias. En los siguientes apartados veremos las características más importantes de este tipo de dispositivos así como los métodos utilizados para llevar a cabo todo el tema de tratamiento de imágenes y cómo a partir de dos imágenes podemos llevar a cabo una reconstrucción 3D completa.

3.1. Adquisición de imágenes

3.1.1. Smartphone

La introducción del Smartphone en la telefonía móvil ha supuesto toda una revolución. En España según un informe realizado por IAB Spain [40], [41], la penetración del Smartphone alcanzó el 80% en el año 2013, siendo Android el sistema operativo más utilizado con casi el 80% de cuota de mercado mundial.

Figura 10. Penetración Smartphone. Fuente: [41]

Figura 11. Cuota de mercado por fabricantes. Fuente: [41]

Este tipo de dispositivos reúnen en un sólo terminal las características de un teléfono móvil convencional en cuanto a posibilidades de llamada y tienen una potencia similar a un ordenador personal, lo que permite no sólo navegar por webs, ver correos electrónicos o acceder a las Redes Sociales mediante conexión Wi-Fi como se podría hacer desde cualquier ordenador preparado para ello, sino que también se puede acceder a todo ello mediante redes de datos móviles, lo que da a estos dispositivos una gran movilidad, además de ser más pequeños y manejables. También hay que añadir la inclusión de serie de diferentes tipos de hardware como cámaras o sensores que no vienen incluidos en un ordenador medio.

Una de las características que definen a este tipo de dispositivos es el uso que hacen de las aplicaciones móviles, que son programas informáticos diseñados para ser ejecutados en smartphones, tabletas u otro tipo de dispositivos móviles. La forma general para acceder a ellas es a través de las plataformas de distribución que cada compañía propietaria de los sistemas operativos móviles tiene.

Ya que Android tiene casi una cuota del mercado del 80% y sus requisitos para programar y trabajar son mínimos, como veremos, será el sistema operativo elegido para el desarrollo y prueba de la presente aplicación.

3.1.2. Android

Android es un sistema operativo inicialmente pensado para teléfonos móviles, pero que posteriormente se ha ido usando en tabletas y otros tipos de dispositivos como pueden ser electrodomésticos, televisores o relojes. Lo que lo hace diferente es que está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma.

Utiliza la máquina virtual Dalvik, que es una implementación de Google de la máquina virtual Java optimizada para dispositivos móviles, es decir, para dispositivos de baja potencia y poca memoria.

Con Android se persigue dar al desarrollador control total sobre todas las funcionalidades que pueden ofrecer los dispositivos móviles (llamadas, mensajes uso de sensores, cámaras, videojuegos, GPS, Wi-Fi, Bluetooth, etc) para poder crear así cualquier tipo de aplicación.

3.1.2.1. Desarrollo de aplicaciones en Android

Para empezar con el desarrollo de aplicaciones en Android es importante conocer cómo está estructurado este sistema operativo. La arquitectura de Android está formada por cuatro capas (figura 12). Cada una de estas capas utiliza servicios ofrecidos por las anteriores, y ofrece a su vez los suyos propios a las capas de niveles superiores, permitiendo al desarrollador acceder a las capas más bajas mediante el uso de librerías para no tener que programar a bajo nivel.

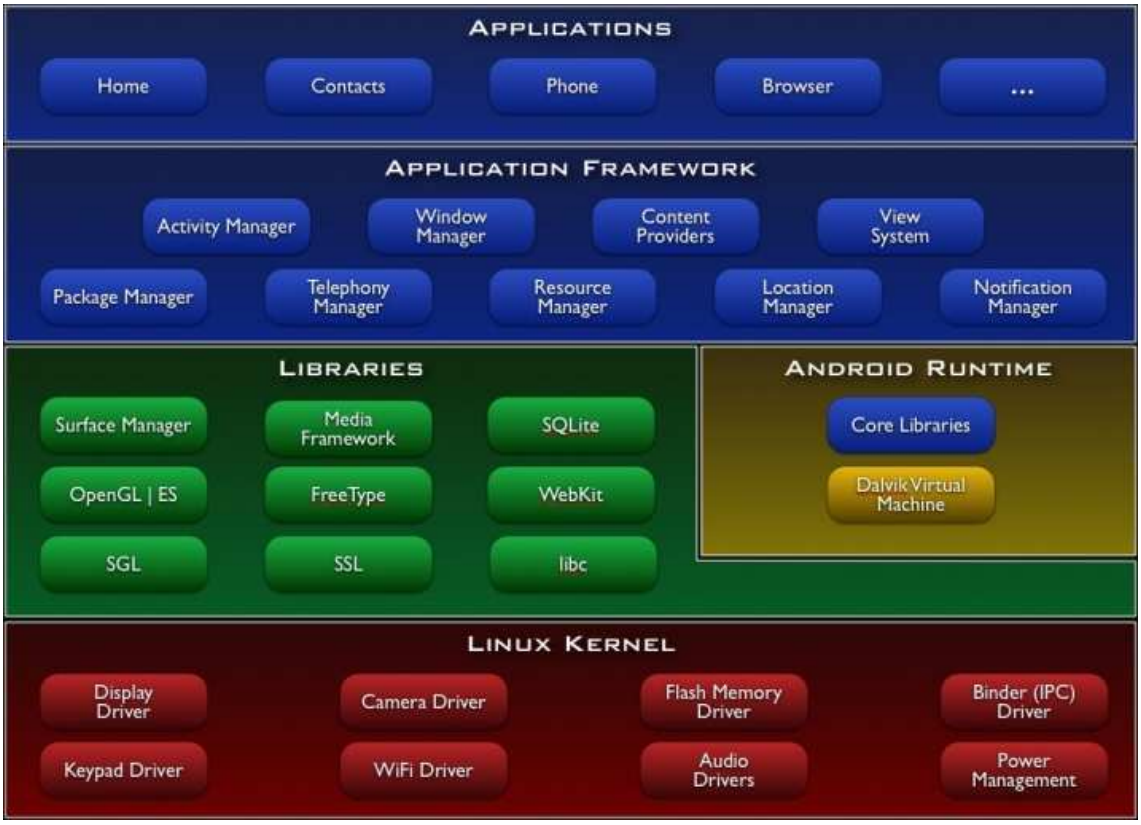


Figura 12. Arquitectura de Android. Fuente: http://elinux.org/Android_Architecture

A continuación se explica brevemente el uso de cada capa de abajo a arriba.

Kernel de Linux. Android utiliza el núcleo de Linux 2.6 como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa proporciona servicios como la seguridad, el manejo de memoria, el multiproceso, la pila de protocolos y el soporte de los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes. Siempre que un fabricante incluye un nuevo elemento de hardware, lo primero que se debe realizar para que pueda ser utilizado desde Android es crear las librerías de control o drivers necesarios dentro de este kernel de Linux embebido en el propio Android.

Runtime de Android. Las aplicaciones se ejecutan mediante la máquina virtual Dalvik, que es una creación de Google ante las limitaciones que tenían los dispositivos móviles (poca memoria y procesador limitado), que trabaja con ficheros ejecutables .dex para facilitar la optimización de recursos. Además está basada en registros. Cada aplicación corre en su propio proceso Linux con su propia instancia de la máquina virtual Dalvik. Delega al kernel de Linux algunas funciones como *threading* y el manejo de la memoria a bajo nivel.

También se incluye en el *Runtine de Android* el "core libraries" con la mayoría de las librerías disponibles en el lenguaje Java.

Librerías. Están escritas en C o C++ y compiladas para la arquitectura hardware específica del teléfono. Entre las librerías más importantes ubicadas aquí, se pueden encontrar las siguientes:

- Librería libc: Incluye todas las cabeceras y funciones según el estándar del lenguaje C.
- Librería Surface Manager: Es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.
- OpenGL/SL y SGL: Representan las librerías gráficas. OpenGL/SL maneja gráficos en 3D y permite utilizar, en caso de que esté disponible en el propio dispositivo móvil, el hardware encargado de proporcionar gráficos 3D. SGL proporciona gráficos en 2D. Se pueden combinar ambos tipos de gráficos.
- Librería Media Libraries: Proporciona todos los códecs necesarios para el contenido multimedia soportado en Android (vídeo, audio, imágenes estáticas y animadas, etc).
- FreeType: Permite trabajar con distintos tipos de fuentes.
- Librería SSL: Posibilita la utilización de dicho protocolo para establecer comunicaciones seguras.
- Librería SQLite: Creación y gestión de bases de datos relacionales.
- Librería WebKit: Proporciona un motor para las aplicaciones de tipo navegador y forma el núcleo del actual navegador incluido por defecto en la plataforma Android.

Framework de Aplicaciones. Representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación. Toda aplicación que se desarrolle para Android, ya sean las propias del dispositivo, las desarrolladas por Google o terceras compañías, o incluso las que el propio usuario cree, utilizan el mismo conjunto de API y el mismo "framework", representado por este nivel.

Entre las API más importantes cabe destacar:

- Activity Manager: gestiona el ciclo de vida de las aplicaciones en Android y proporciona un sistema de navegación entre ellas.
- Window Manager: Gestiona las ventanas de las aplicaciones y utiliza la librería Surface Manager.
- Telephone Manager: Incluye todas las API vinculadas a las funcionalidades propias del teléfono (llamadas, mensajes, etc.).
- Content Provider: Permite a cualquier aplicación compartir sus datos con las demás aplicaciones de Android (como los contactos).
- View System: Proporciona un gran número de elementos para poder construir interfaces de usuario (GUI).
- Location Manager: Posibilita a las aplicaciones la obtención de información de localización y posicionamiento.
- Notification Manager: permite a las aplicaciones mostrar alertas personificadas en la barra de estado.
- XMPP Service: Colección de API para utilizar este protocolo de intercambio de mensajes basado en XML.

Aplicaciones. Este nivel contiene tanto las incluidas por defecto de Android como aquellas que el usuario vaya añadiendo posteriormente, ya sean de terceras empresas o de su propio

desarrollo. Todas estas aplicaciones utilizan los servicios, las API y librerías de los niveles anteriores.

Las aplicaciones se encuentran aisladas unas de otras gracias al concepto caja de arena o sandbox que hereda de Linux y que proporciona un entorno seguro de ejecución. Cada aplicación tiene una serie de permisos que restringen su rango de actuación e interacción con otras aplicaciones. Para que varias aplicaciones interactúen entre sí o con elementos del sistema es mediante la declaración explícita de un permiso que autorice a llevar a cabo una determinada acción habitualmente prohibida.

Las aplicaciones se desarrollan habitualmente en el lenguaje Java con Android SDK (Android Software Development Kit) [42], pero están disponibles otras herramientas de desarrollo, incluyendo un kit de Desarrollo Nativo (Android NDK) [42] para aplicaciones o extensiones en C o C++, Google App Inventor, que es un entorno virtual para desarrolladores novatos, aunque aún se encuentra en fase beta [45].

Las aplicaciones están comprimidas en formato apk, que se puede instalar sin dificultad desde cualquier explorador de archivos en la mayoría de dispositivos.

Para desarrollar la aplicación se usara el Android Developer Tools (ADT) Bundle proporcionado por Google en versión 64 bits, aunque también existe una versión para procesadores de 32 bits, que se puede descargar desde su web [42], y que incluye las siguientes herramientas:

- El entorno de desarrollo Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- La última versión de la plataforma Android
- La última versión del emulador para ordenador

Existen otros entornos de desarrollo diferentes, como por ejemplo Android Studio [42], basado en IntelliJ IDEA, pero aún está en fase de desarrollo y puede contener algún bug, por lo que se usará el entorno de Eclipse.

Las aplicaciones pueden probarse de diversas formas mientras se van desarrollando, directamente en un **teléfono móvil**, activando la opción 'Fuentes desconocidas' ubicada en 'Ajustes->Aplicaciones' y la opción 'Depuración USB' que se encuentra en 'Ajustes-Aplicaciones->Depuración', o mediante el **emulador** [42] creando un dispositivo virtual Android (AVD) proporcionado por el SDK de Google, en el que se pueden configurar y tener varios dispositivos virtuales. Se pueden emular tanto dispositivos reales, como por ejemplo un Nexus 7 o bien crear uno genérico en función de la versión del sistema operativo en el que se quiera testear la aplicación, el tamaño de la pantalla del dispositivo, memoria RAM, si acepta o no tarjeta de memoria externa, etc.

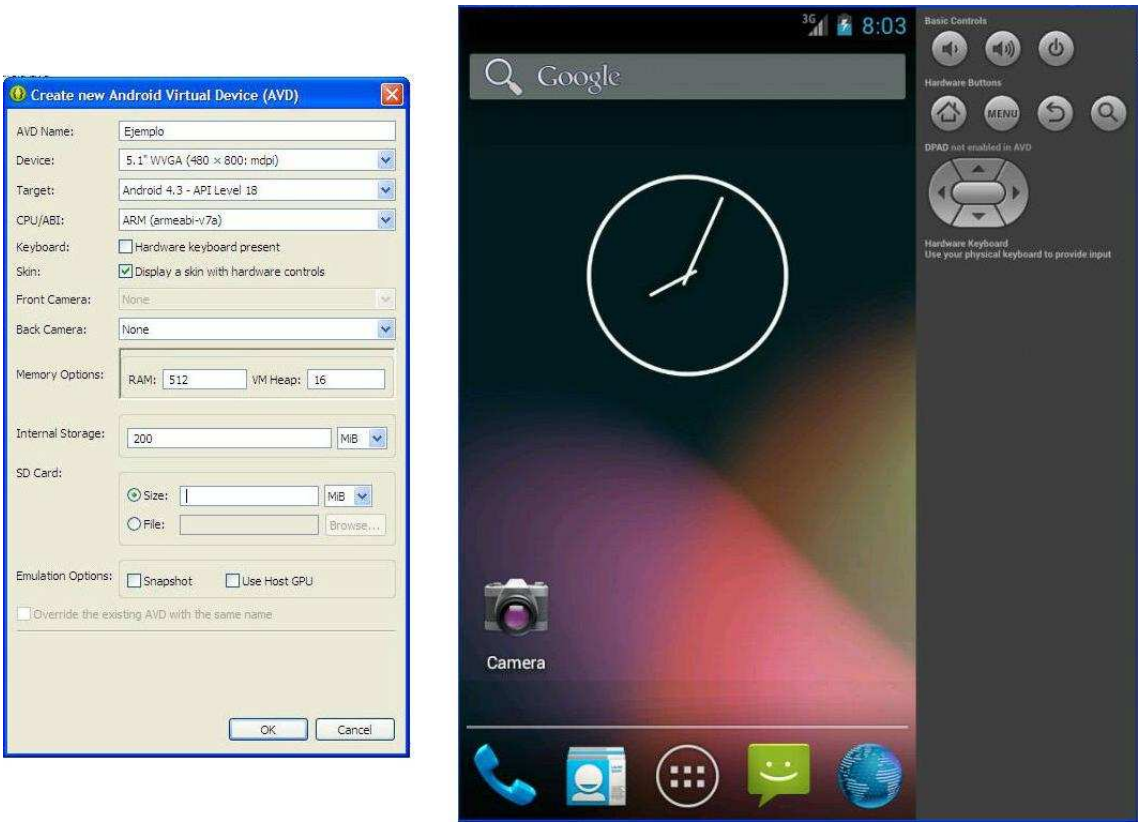


Figura 13. Izquierda: Ejemplo de configuración del emulador. Derecha: Ejemplo de ejecución del emulador con versión Android 4.2.2

En el emulador se puede utilizar la webcam del ordenador si está disponible e incluso existe software para emular los sensores disponibles en un dispositivo real, aunque esto último resulta laborioso y al final habrá que realizar dos códigos, uno para el emulador y otro para un terminal real.

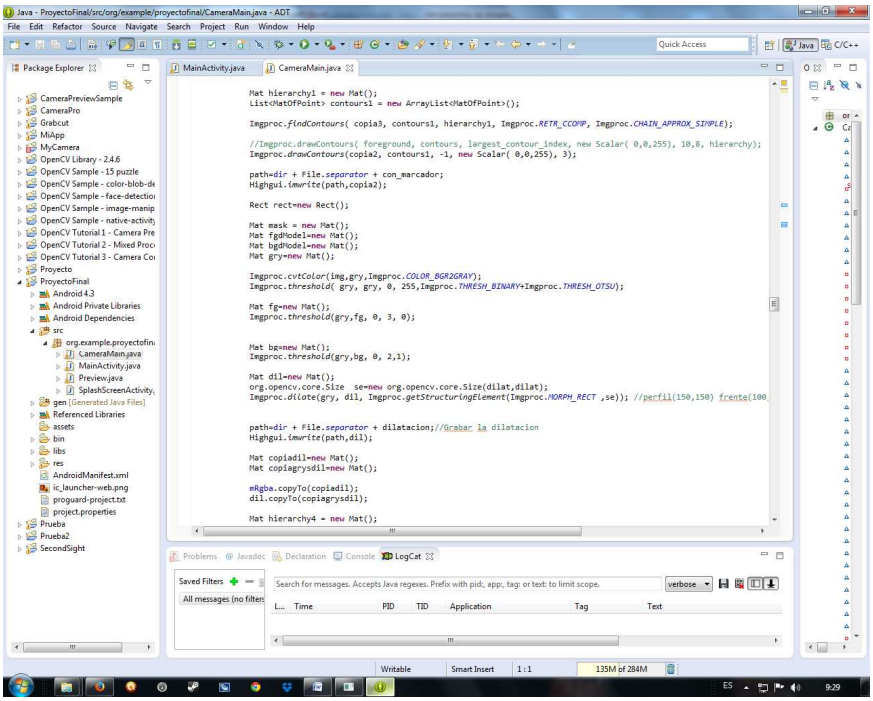


Figura 14. Captura de la interfaz de eclipse

El mayor inconveniente que tiene Android a la hora de desarrollar aplicaciones es la gran cantidad de versiones que hay de este sistema operativo, como se puede ver en la figura 11, lo que supone que si se quiere abarcar la mayor cantidad de dispositivos en el mercado seguramente no puedas hacer uso de todas las características introducidas en las últimas versiones o características que por un motivo u otro no han tenido continuidad en versiones posteriores o directamente se han eliminado, como por ejemplo el sensor para medir la orientación de un dispositivo al obtener los datos directamente del acelerómetro y del sensor de campo geomagnético requiere mucho procesado y su uso está 'desaprobado' desde la versión 2.2 (nivel de API 8) y no ha tenido continuidad en las siguientes versiones.

Al comienzo de crear la aplicación se puede seleccionar desde que versión a que versión de Android se desea que esta funcione, pero eso se puede cambiar en cualquier momento desde el manifiesto de la aplicación.

Version	Codename	API	Distribution
2.2	Froyo	8	1.0%
2.3.3 - 2.3.7	Gingerbread	10	16.2%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	13.4%
4.1.x	Jelly Bean	16	33.5%
4.2.x		17	18.8%
4.3		18	8.5%
4.4	KitKat	19	8.5%

Figura 15. Distribución versiones Android. Fuente: http://www.xatakamovil.com/sistemas-operativos/la-distribucion-de-versiones-android-en-mayo-kit-kat-sube-jelly-bean-down

3.1.2.2. Ciclo de vida de las aplicaciones en Android

En Android es el sistema operativo el que se encarga de controlar el ciclo de vida de cada aplicación, el usuario no es consciente de todo este proceso. Cada aplicación se ejecuta en su propio proceso Linux y es el propio sistema operativo el que se encarga de gestionar su ejecución y decidir cuándo hay que parar una aplicación en función de los recursos disponibles.

Si el sistema destruye el proceso de una aplicación y esta es requerida por el usuario más adelante, se vuelve a crear de nuevo el proceso perdiéndose el estado que tenia la aplicación anteriormente, por lo que la tarea de almacenar los estados de las actividades recae en la figura del desarrollador.

Una aplicación va a estar formada por varios elementos básicos de visualización, que son conocidos como actividades, que son extensiones de la clase Activity. Son estas actividades las que se encargan de controlar el ciclo de vida de la aplicación, ya que el usuario lo que cambia es de actividad, no de aplicación. El sistema se guarda en una pila las actividades previamente mostradas.

Las actividades pueden encontrarse en cuatro estados:

- **Running**. La actividad es visible y tiene el foco, es decir, que está arriba del todo en la pila y el usuario puede interactuar con ella.
- **Paused**. La actividad es visible pero no tiene el foco. Por ejemplo cuando se abre un dialogo encima de la pantalla.
- **Stopped**. Cuando una actividad ya no es visible. Es en este estado cuando se recomienda guardar el estado de la interfaz de usuario, preferencias, etc.
- **Destroyed**. La actividad termina al llamar al método finish() o el propio sistema operativo ha decidido matarla y sale de la pila de actividades.

El desarrollador puede controlar en cada momento que eventos se producen en cada estado que pueden ser capturados por ciertos métodos de la actividad. A continuación se muestran estos eventos y una breve explicación de los mismos.

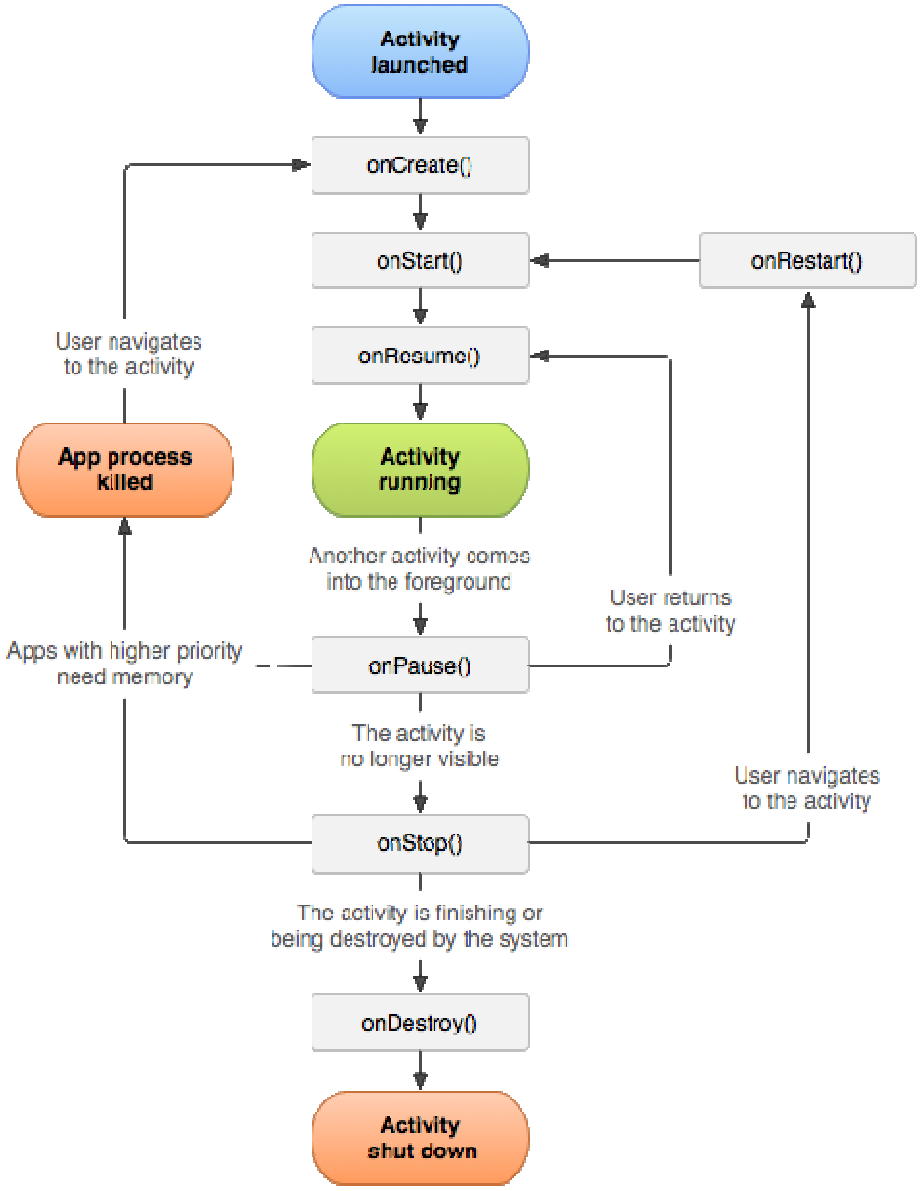


Figura 16. Ciclo de vida de una actividad. Fuente: [42]

onCreate(). Se llama en la creación de la actividad y es la parte en la que se realizan todo tipo de inicializaciones como la creación de la interfaz de usuario. Puede recibir información de estado de instancia por si la actividad ha sido destruida y vuelta a crear.

onStart(). Indica que la actividad está a punto de ser mostrada al usuario.

onResume(). Se llama cuando la actividad va a comenzar a interactuar con el usuario. Es el lugar indica para lanzar animaciones y música.

onPause(). Indica que la actividad está a punto de ser lanzada en segundo plano. Es el lugar indicado para parar animaciones o música y almacenar datos.

onStop(). La actividad ya no va a ser visible para el usuario (está en segundo plano). Si la memoria es insuficiente es posible que la actividad se destruya sin pasar por este estado.

onRestart(). Indica que la actividad va ser llamada otra vez después de pasar por onStop().

onDestroy(). La actividad va a ser destruida y sus recursos liberados, por ejemplo cuando se pulsar el botón volver o se llama al método finish(). Como en onStop(), si la memoria no es suficiente es posible que se destruya la actividad sin pasar por este método.

3.1.2.3 Estructura de un proyecto Android

Un proyecto en Android está formado principalmente por un descriptor de la aplicación (AndroidManifest.xml), el código fuente en Java y una serie de ficheros de recursos. Cada elemento se almacena en una carpeta específica. Para ver la estructura de un proyecto Android se tomará como ejemplo el del propio proyecto.

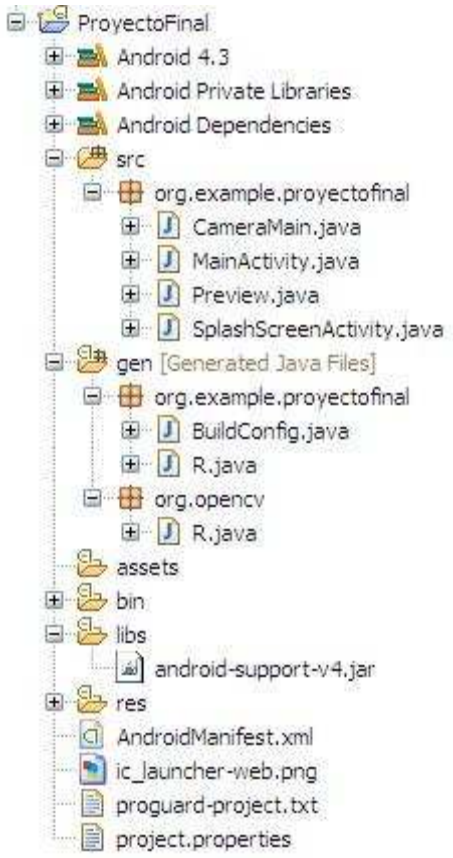


Figura 17. Estructura de la aplicación

Android x.x: Código JAR, el API de la versión de Android seleccionada.

Android Private Libraries y Android Dependencies: Librerías asociadas al proyecto, por ejemplo en este caso la librería de OpenCV.

src: Carpeta que contiene el código fuente de la aplicación.

gen: Contiene el código generado automáticamente por el SDK al compilar el proyecto. Estos ficheros nunca hay que modificarlos.

> **BuildConfig.java**: Almacena la constante DEBUG para saber si la aplicación está en f ase de desarrollo.
>
> **R.java**: Asocia los recursos de la aplicación con identificadores. OpenCV define su propia clase R.java.

assets: Puede contener una serie de ficheros auxiliares o carpetas que podrán ser utilizados por la aplicación (ficheros de configuración, de datos,

fuentes, …). Nunca se modifica el contenido de los ficheros ni se les asocia identificador.

bin: Contiene los elementos compilados de la aplicación y otros ficheros auxiliares. Almacena el fichero .apk, que contiene la aplicación lista para instalar.

libs: Contiene librerías auxiliares en formato .JAR que se quieran utilizar en los proyectos. Se añade una librería de forma automática (android-support) que añade nuevas funcionalidades que no aparecían en el nivel de API 4.

Figura 18. Más información de la estructura de la aplicación

res: Contiene los recursos usados por la aplicación. A estas carpetas se les pueden añadir un sufijo (por ejemplo layout-land, layout-xlarge) si queremos que el layout de nuestra aplicación se muestre de forma diferente si el dispositivo esta en landscape o para pantallas grandes como tabletas o si queremos que la aplicación se puede mostrar en otro idioma (values-en para ingles), etc.

drawable: Almacena los ficheros de imagen. Se añade un sufijo en función de la resolución y densidad de la pantalla.

layout: Contiene los ficheros XML con las vistas de la aplicación.

menu: Contiene los ficheros XML que forman los menús de la aplicación.

values: Ficheros XML para indicar valores del tipo string, color o estilo. Permite cambiar los valores sin necesidad de ir al código fuente.

anim: Ficheros XML con animaciones Tween.

animator: Ficheros XML con animaciones de propiedades.

xml: Otros ficheros XML de datos requeridos.

raw: Contiene recursos adicionales en formato distinto a XML.

AndroidManifest.xml: Fichero en el que se indican las actividades, intenciones, permisos requeridos, versión, etc.

ic_launcher-web.png: Icono de la aplicación de gran tamaño para ser usado en páginas web.

proguard-project.txt: Fichero de configuración de la herramienta ProGuard que permite optimizar y ofuscar el código generado.

default.properties: Fichero generado automáticamente por el SDK que no hay que modificar. Sirve para comprobar la versión del API y otras características al instalar la aplicación en el terminal.

3.1.2.4. Componentes de una aplicación.

Hay una serie de componentes en Android que son esenciales a la hora de crear una aplicación. En este apartado se verán de forma general los componentes más importantes.

View. Son los elementos que componen la interfaz de usuario, como un botón o entrada de texto. Aunque se pueden definir usando Java (pues estos elementos descienden de la clase View), lo habitual es definirlos en un fichero XML y dejar que el sistema cree estos objetos.

Layout. Conjunto de vistas agrupadas de una determinada forma. Hay diferentes tipos de layouts en función de la forma en que queramos agrupar las vistas, como por ejemplo de forma lineal o en cuadricula. También se pueden definir usando Java, pero es más cómodo utilizar XML.

Activity. Es el componente básico que permite crear componentes que interaccionan con el usuario. Su función principal es la de crear la interfaz de usuario. Una aplicación suele tener varias actividades para realizar este propósito. Todas las actividades tienen que descender de la clase Activity.

Intent. Componente básico que permite el envió de mensajes para interconectar componentes de la misma o de distintas aplicaciones. Las intenciones se utilizan para arrancar actividades o enviar eventos a múltiples destinatarios. Se pueden lanzar **explícitamente,** indicando mediante código que componente se quiere lanzar, o **implícitamente**, dejando que sea la plataforma la que elija el componente apropiado para su ejecución.

Service. Es un proceso que se ejecuta en segundo plano, como los demonios en Unix o los servicios en Windows.

Broadcast Receiver. Recibe y reacciona ante mensajes tipo broadcast, como por ejemplo mensajes del sistema como batería baja, llamada de entrada**,** o generados por una aplicación.

Content Provider. Para poder acceder a los datos de otras aplicaciones como por ejemplo la lista de contactos sin comprometer la seguridad del sistema de ficheros.

3.2. Tratamiento de imágenes

Para el tratamiento de imágenes Android no ofrece al desarrollador herramientas muy avanzadas, por eso hay que incluir librerías externas.

3.2.1. OpenCV

OpenCV (Open Source Computer Vision Library) [46] es una biblioteca libre de visión artificial que incluye más de 2500 algoritmos optimizados, que van desde algoritmos clásicos hasta los más avanzados en visión por ordenador y algoritmos de aprendizaje automático. Estos algoritmos se pueden utilizar para detectar y reconocer rostros, identificar objetos, clasificar las acciones humanas en los videos, los movimientos de cámara de la pista, los objetos en movimiento de la pista, extraer modelos 3D de objetos, producen nubes de puntos 3D a partir de cámaras estéreo, unir imágenes para producir una alta resolución imagen de una escena completa, encontrar imágenes similares de una base de datos de imágenes, eliminar los ojos

rojos de las imágenes tomadas utilizando el flash, seguir los movimientos de los ojos, reconocer el paisaje y establecer marcadores para revestirlo de la realidad aumentada, etc.

Tiene interfaces para C++, C, Python, Java y Matlab y es multiplataforma, con soporte además de Android para GNU/Linux, Mac Os y Windows. Actualmente se están desarrollando interfaces con todas las funciones CUDA y OpenCL.

De todas las funciones que nos ofrece esta librería, nos vamos a centrar en las opciones que ofrece para segmentar imágenes, en concreto dos algoritmos: Watershed y GrabCut.

3.2.2. Watershed

Este algoritmo, diseñado principalmente por Serge Beucher and Fernand Meyer [47], se trata a las imágenes como si fueran una superficie topográfica, es decir, se consideran dos dimensiones espaciales y la intensidad para tratar la altura (a más intensidad, picos más altos). Se consideran tres tipos de puntos:

1. Puntos que pertenecen a mínimos locales
2. Puntos en los que si se coloca una gota de agua esta caería con certeza en un mínimo local
3. Puntos en los que el agua caería con la misma probabilidad en más de un mínimo.

Para un mínimo local, los puntos que satisfacen la condición 2 forman la cuenca de un mínimo y los que satisfacen la condición 3 forman las líneas de cresta en la superficie topográfica y son llamados líneas de división.

El principal objetivo de este algoritmo es encontrar las líneas divisorias que separan las diferentes cuencas y realizar así la segmentación de la imagen. Lo veremos con el siguiente ejemplo.

Partimos de una imagen en tonos grises (figura 19 izquierda), siendo la imagen de la derecha su representación topográfica y la altura de las “montañas” el valor de intensidad de la imagen original.

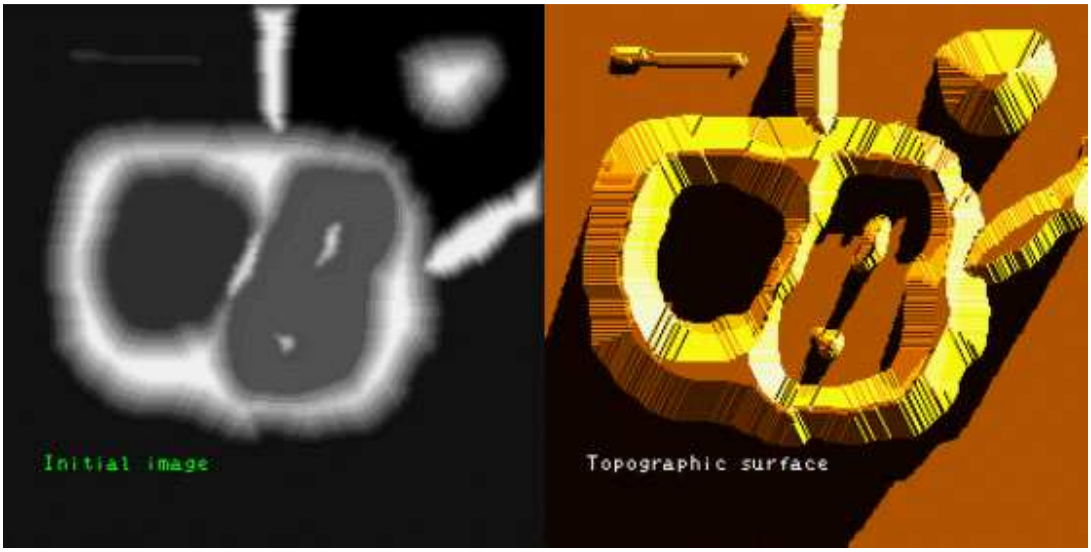


Figura 19. Ejemplo segmentación watershed. Fuente: http://cmm.ensmp.fr/~beucher/wtshed.html

Cada mínimo local se va llenando de agua a una velocidad uniforme. Llegará un momento en el que el agua se desbordara de una cuenca a otra, momento en el cual se construirá un dique en la zona desbordada para impedir el paso del agua. Este proceso continua hasta que se alcanza el máximo nivel de inundación, que suele ser el máximo nivel de intensidad de la imagen, obteniéndose la segmentación deseada.

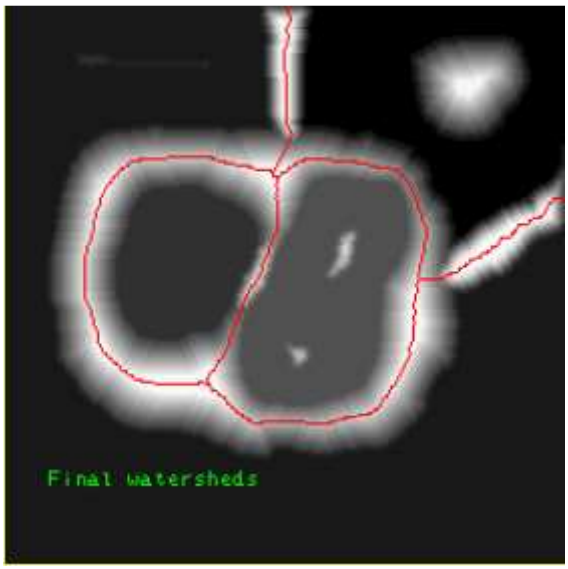


Figura 20. Resultado final de watershed. Las marcas rojas indican la detección de la figura. Fuente: http://cmm.ensmp.fr/~beucher/wtshed.html

Pero aplicar directamente esta aproximación suele dar importantes errores de sobre segmentación debido a ruido y otras irregularidades en la imagen.

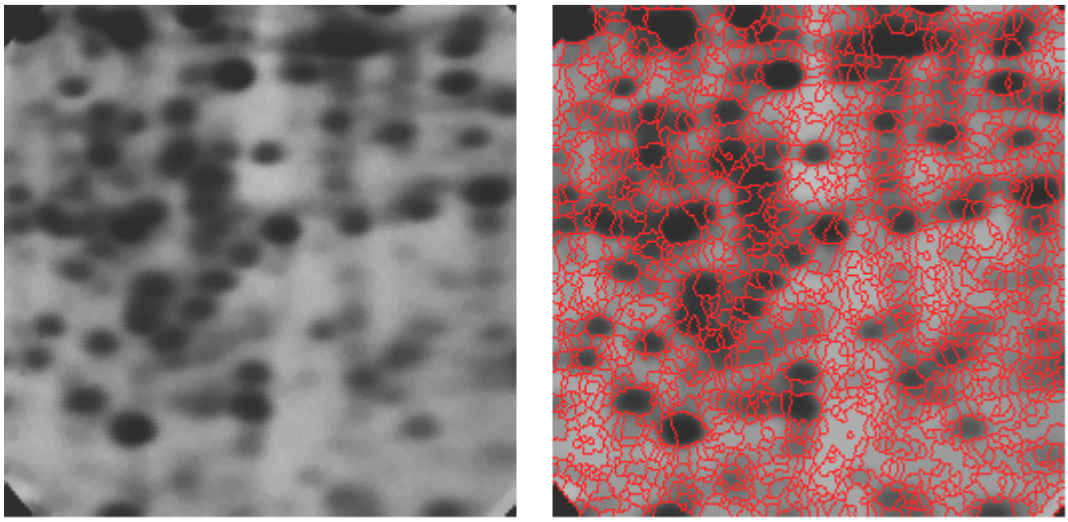


Figura 21. Ejemplo de sobre segmentación del Imagen del gel de electroforesis. Fuente: http://cmm.ensmp.fr/~beucher/wtshed.html

Para prevenir la sobre-segmentación se utilizan marcadores introducidos manualmente indicando lo que será el foreground y el background de la imagen antes del aplicar el algoritmo.

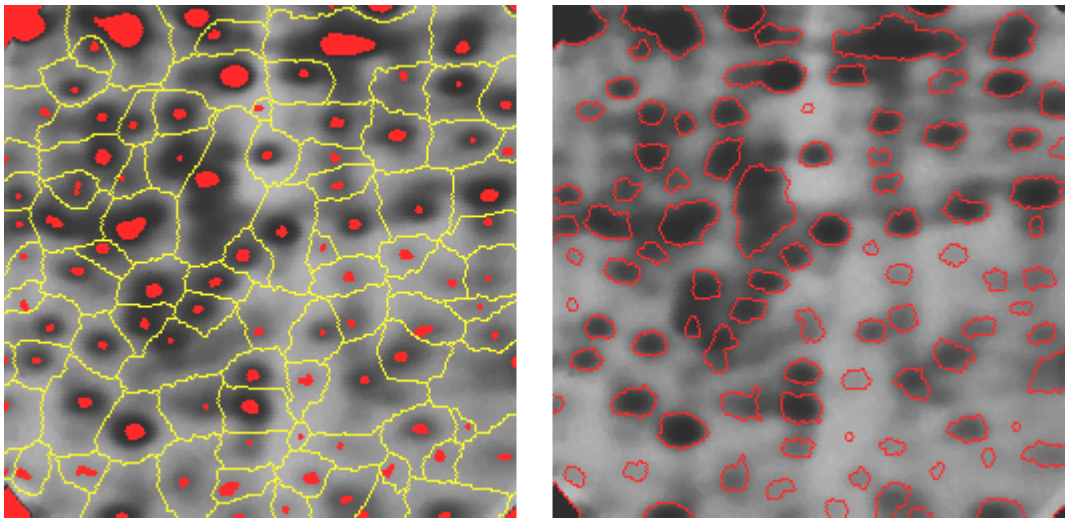


Figura 22. Izquierda: Ejemplo de marcadores. En rojo se marca el foreground y en amarillo el background. Derecha: Resultado de aplicar watershed. Fuente: http://cmm.ensmp.fr/~beucher/wtshed.html

3.2.3. Grab-Cut

Este algoritmo, diseñado por Carston Rother, Vladimir Kolmogorov y Andrew Blake [48], supone una mejora del método Graph-Cut [Boykov and Jolly 2001; Greig et al. 1989] para segmentación de imágenes, que mejora a este último en tres aspectos. Primero introduce una versión iterativa más potente de la optimización. Segundo, la fuerza del algoritmo iterativo reduce la interacción por parte del usuario y tercero, introduce un algoritmo para obtener mejor contraste con los bordes de la imagen segmentada.

A continuación veremos el funcionamiento de Graph-Cut, que es la base de GrabCut, y las mejoras que introduce esta versión.

En Graph-Cut se parte de una imagen en blanco y negro y para conseguir la segmentación se construye un grafo, donde cada nodo de representa un pixel de la imagen y además se construyen dos nodos especiales, llamados Sink y Source, que representan a lo que será background y foreground respectivamente. Cada nodo se conecta a estos nodos especiales midiendo la probabilidad de que sean foreground o background. Estas probabilidades se obtienen a partir de las zonas de la imagen que el usuario ha marcado de diferente color como seguro foreground o seguro background antes de empezar el proceso de segmentación.

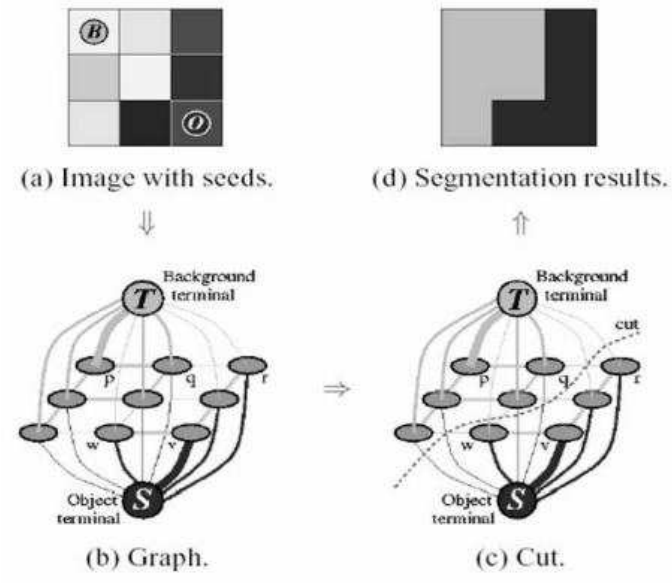


Figura 23. Ejemplo visual de Graph-Cut. Fuente: [50]

En el siguiente paso se conecta cada nodo con sus vecinos mediante unos pesos determinados por la información sobre los bordes de la imagen. El valor de estos pesos indica la similitud que hay entre ellos, siendo este valor mayor cuanta más similitud haya entre ellos (por ejemplo el color del pixel), indicando la presencia de bordes en la imagen.

Una vez acabado el grafo con sus pesos, se usan algoritmos de Min-Cut/ Min-Flow para determinar el valor de mínimo corte, separando por un lado los pixeles unidos al nodo Source (foreground) del nodo Sink (background) y completando la segmentación.

Figura 24. Ejemplo de Graph-Cut. Fuente: [48]

En Grabcut se pasa a trabajar con imágenes en color, y la segmentación se consigue con los siguientes pasos:

1. Primero el usuario marca con un rectángulo la zona a segmentar en la imagen, dejando dentro todo lo que se quiera segmentar, marcándose los pixeles que quedan fuera como seguro background y la parte de dentro los pixeles se marcan como desconocidos.

2. El ordenador hace una marcación inicial como foreground o background en función de los datos que le pasamos.
3. Se usa un GMM (Gaussian Mixture Model) para modelizar el foreground y el background. El uso de GMMs se debe a que estamos en el espacio de color, por lo que usar histogramas como en Graph-Cut es impracticable.
4. Dependiendo de los datos pasados, los modelos GMM aprenden y van marcando los pixeles desconocidos como probable foreground o probable background en función de la relación que tengan con los pixeles marcados por el usuario.
5. Una vez se tiene la relación de los pixeles se construye un grafo y se siguen los mismos pasos que en Graph-Cut para separar los pixeles en foreground y en background.
6. Se vuelven a repetir los pasos hasta que la clasificación de los pixeles converge.
7. Se aplica el algoritmo de border-matting desarrollado.

En muchos casos sólo con marcar el rectangulo ya se podría conseguir una buena segmentación y acabaría la parte del usuario, pero hay casos en que los que los colores de la zona a segmentar no se distinguen demasiado del fondo y la segmentación no es suficiente y desaparece parte del foreground o aparece parte del background. Para estos casos también se da la opción de incluir marcadores propios como en Graph-Cut, es decir, marcar una zona como seguro foreground y otra como seguro background, volviendo a repetirse el proceso desde el paso 2.

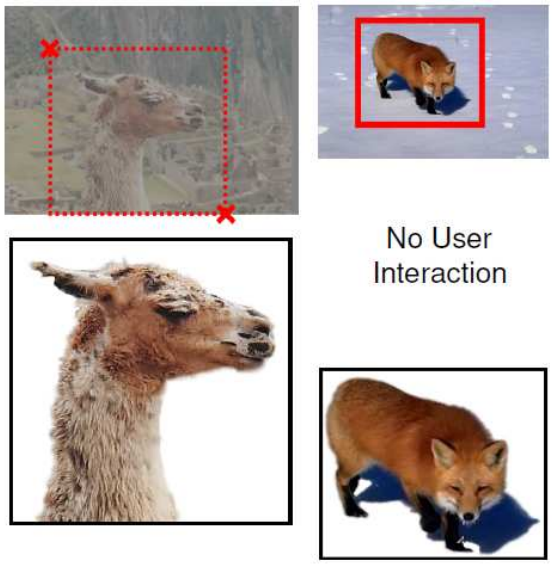


Figura 25. Ejemplo de Grab-Cut con segmentación automática. Fuente: [48]

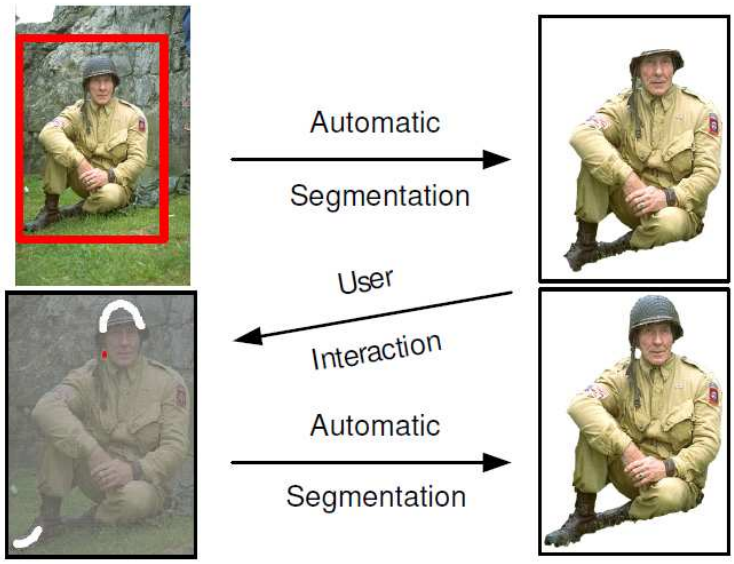


Figura 26. Ejemplo de GrabCut con marcación de foreground (blanco) y background (rojo) por parte del usuario. Fuente: [48]

3.3. Obtención de modelo 3D a partir de los perfiles 2D

Como una vez calculados los contornos de las siluetas se van a hacer pruebas con los algoritmos desarrollados en el IBV que luego se utilizaran cuando todo el proyecto esté operativo, a saber, obtención de los contornos, envío del fichero al servidor, reconstrucción 3D, envío de vuelta al usuario, uso para copra online, etc, en este apartado veremos por encima (para calmar un poco la curiosidad) cómo a partir de los contornos de frente y de perfil calculados por la aplicación desarrollada en este proyecto se puede reconstruir un modelo corporal 3D.

Para la reconstrucción del modelo 3D a partir de los perfiles 2D nos vamos a basar en el método descrito en el artículo “PCA-based 3D Shape Reconstruction of Human Foot Using Multiple Viewpoint Cameras” [49] y desarrollado por Edmée Amstutz, Tomoaki Teshima, Makoto Kimura, Masaaki Mochimaru y Hideo Saito. Aunque este se basa en la reconstrucción de un pie, los pasos son muy parecidos. A continuación se explicará brevemente su funcionamiento.

Primero desde una base de datos se define un modelo 3D inicial y se determinan los parámetros más importantes que caracterizan un cuerpo mediante el uso del método análisis de componentes principales (PCA).

Con la calibración de la cámara obtenida a la hora de hacer las fotografías, se hace una transformación afín para que encajen los centros de gravedad del modelo inicial 3D y los de las fotografías capturadas, que será en 2D.

Una vez calculado el centro de gravedad se prosigue de la siguiente manera: el modelo 3D se proyecta en las imágenes tomadas, usando los resultados de la calibración de la cámara. Luego se calcula el error entre el modelo proyectado y el cuerpo real y utilizando técnicas de optimización se minimiza el error cambiando el modelo inicial y repitiendo los pasos anteriores. Modificar el modelo inicial consiste en aplicar traslación, rotación y escala y cambiar los parámetros más importantes de forma definidos por el análisis de componentes principales.

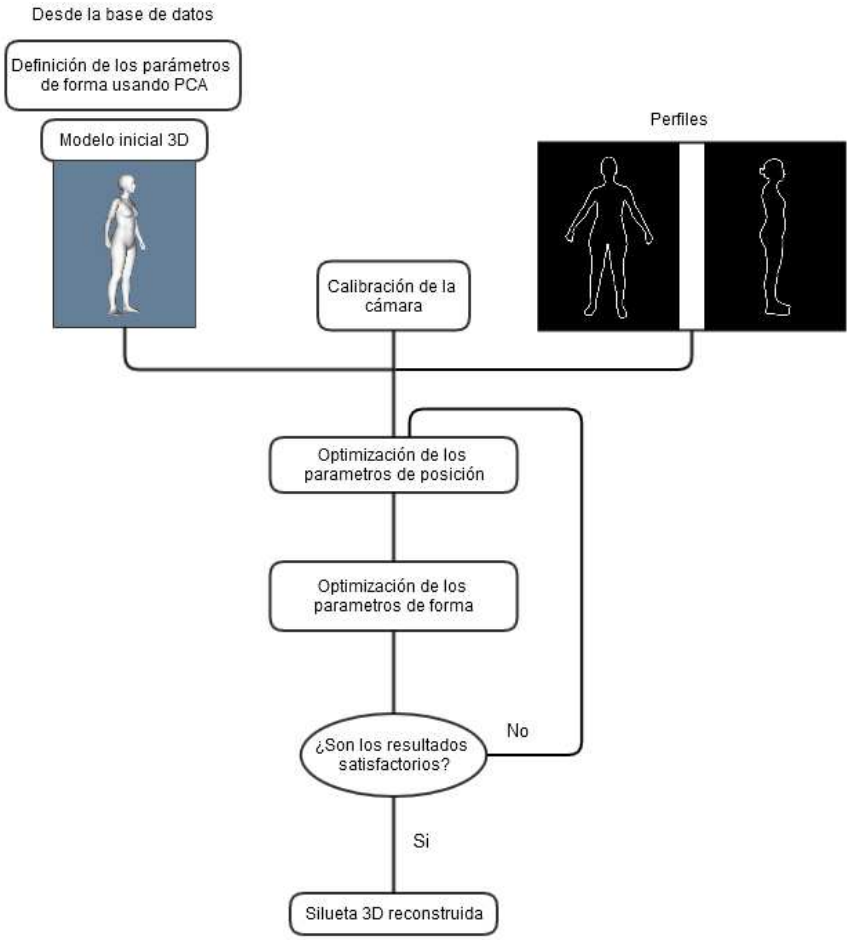


Figura 27. Diagrama de flujo de la reconstrucción 3D

Una vez reconstruido el modelo 3D, mediante librerías propias desarrolladas en el IBV, se calculan las medidas antropométricas necesarias que pueden servirtanto para la confección de ropa, para un futuro probador de ropa virtual o para futuras aplicaciones que se puedan desarrollar.

Capítulo 4: Desarrollo de la aplicación

En este apartado no se pretende explicar paso a paso todo el desarrollo de la aplicación, sino explicar los elementos más importantes utilizados para que el lector tanto si esta versado en la programación en Android como si no, tenga una idea de los elementos usados y del funcionamiento que se le da a la librería OpenCV.

La aplicación se puede dividir en cuatro partes principales, una primera parte que servirá como tutorial para el usuario, una segunda en la que se utilizará la cámara y los sensores del dispositivo, la tercera que sería el uso de la librería OpenCV y una cuarta que será el almacenamiento de resultados. Cada parte se explicará en los siguientes apartados.

4.1. Diseño de la interfaz de usuario

La interfaz de usuario no es definitiva y se cambiará según la evolución que siga la aplicación después de este proyecto. Lo que sí que tendrá en común y que se ha intentado implementar en este proyecto es que antes de empezar a interactuar con la cámara se le dará al usuario un tutorial con una serie de pasos a seguir para que se pueda realizar bien la segmentación así como lo que se irá encontrando conforme vaya avanzando en la aplicación.

Se utilizará para este propósito la vista ViewFlipper [42], que extiende de ViewAnimator, que sirve para animar el paso entre dos o más vistas con solo deslizar el dedo por la pantalla del teléfono hacia la izquierda o la derecha según queramos ir hacia adelante o hacia atrás de la lista de vistas que queramos utilizar, pudiendo de esta forma el usuario, si no ha entendido alguno de los pasos, poder volver atrás.

El tutorial se divide en varios ficheros XML que se le pasan al componente ViewFlipper, que como es lógico se tiene que definir en un XML que será el que le pasaremos a la actividad principal. Los pasos que se muestran en este tutorial serán los siguientes y que se han cumplido en las pruebas realizadas:

- Habitación bien iluminada con pared de fondo blanca y sin objetos detrás.
- Llevar ropa ajustada y de color oscuro para crear contraste con el fondo blanco para hacer más fácil la segmentación.
- Primero se tomará una foto del usuario de frente, mostrando cómo tiene que encajar el usuario en la silueta que saldrá por la pantalla del dispositivo. También se mostrará un ejemplo de buena y mala segmentación para que el usuario sepa qué tiene que esperar, ya que este no tiene por qué saber nada, ni se espera, de tratamiento de imagen.
- Luego se mostrará el resultado de la segmentación y se le preguntará al usuario si el resultado es correcto o no para que puede volver a repetir la foto.
- Después de esto se hará una foto de perfil. Al igual que antes se muestra un ejemplo de cómo tiene que colocarse el usuario en la silueta, así como ejemplos de buena y mala segmentación. También se indica que los brazos tienen que estar rectos y lo más pegados posibles al cuerpo para que no se formen salientes o curvas que deformen el contorno en la zona de la espalda.
- Luego se mostrará el resultado de la segmentación y la posibilidad de volver a repetir si algo sale mal.
- Y antes de empezar el uso de la cámara se le pedirá al usuario una serie de datos (peso, altura y edad).

4.2. Manejo de la cámara

Para manejar la cámara con Android se pueden seguir dos métodos [42]:

Usar una aplicación ya existente mediante una intención. Esta forma serviría si el uso de la cámara fuera una parte poco importante de la aplicación y con un uso básico fuera suficiente.

Controlar directamente la cámara del dispositivo. Para casos en que una aplicación se basa principalmente en esta y se le quiere dar un uso concreto. En este caso habría que construir desde cero todo el proceso, como podría ser mostrar lo que la cámara ve por la pantalla del dispositivo o controlar si hace falta usar flash o el uso del autofoco.

Como en este proyecto el uso de la cámara es esencial ya que hay que usarla de una forma muy específica, usaremos el segundo método.

La clase Camera es la que se usa para configurar los ajustes de captura de imágenes, iniciar/detener la vista previa, tomar fotos y recuperar los marcos para la codificación de video. Esta clase es un cliente para el servicio Camera que gestiona el hardware actual de la cámara.

Para acceder a la cámara del dispositivo se tienen que declarar los permisos correspondientes en el Manifiesto de la aplicación [42], que no es más que un fichero escrito en XML que tienen que tener todas las aplicaciones y que presenta toda la información sobre la aplicación que necesita saber el sistema antes de ejecutar el código. Entre otras cosas se encarga de:

- Nombrar el paquete Java de la aplicación que sirve como identificador único de la aplicación.
- Describe los componentes de la aplicación como son las actividades servicios, broadcast receivers, y proveedores de contenidos de los que está compuesta la aplicación.
- Determina qué procesos serán los anfitriones de componentes de la aplicación.
- Declara que permisos debe tener la aplicación para poder acceder a partes protegidas de la API e interactuar con otras aplicaciones y que permisos tienen que tener estas otras para interactuar con los componentes de la aplicación.
- Se enumeran las clases y otra información de la aplicación que se está ejecutando.
- Declara el mínimo nivel de API requerido para hacer funcionar la aplicación.
- Lista las librerías que la aplicación debe incluir.

En el caso de la cámara en el Manifiesto se deben incluir elementos del tipo <uses-feature> para indicar que características de la cámara vamos a usar, como por ejemplo para decir que queremos hacer uso de esta y la función autofoco, se tendría que incluir lo siguiente:

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />
```

Para tomar fotos desde la página de desarrollares de Android se hace hincapié en las siguientes recomendaciones:

Abrir el objeto Camera: Obtener una instancia del objeto Camera con Camera.open() es el primer paso en el proceso de controlar la cámara directamente.

Llamar a este método lanza una excepción si la cámara ya está en uso por otra aplicación, por lo que hay que hacerlo en un bloque try catch.

```
try {

    mCamera = Camera.open(id);

} catch (Exception e) {
    Log.e(getString(R.string.app_name), "failed to open Camera");

}
```

Desde el API 9 se pueden soportar varias cámaras, por lo que si al método open() no se le pasa argumento se usara la cámara trasera el móvil por defecto.

Crear la clase Camera Preview: Se usa para mostrar por pantalla lo que está viendo la cámara en tiempo real para así que el usuario pueda ver a lo que le hará una foto. Se suele usar una vista SurfaceView para dibujar lo que el sensor de la cámara capta e implementar la interface android.view.SurfaceHolder.Callback que se usa para pasar la imagen desde el hardware de la cámara hasta la aplicación.

Fijar y empezar el preview: Después de instanciar el objeto camera ya se puede llamar al método Camera.startPreview() para empezar a mostrar en la pantalla del dispositivo lo que va captando el sensor.

Es importante configurar bien el tamaño del preview ya que si no el aspect-ratio de lo que se muestre por pantalla será incorrecto, que aunque no afecta a lo capturado por el sensor de la cámara, sí que es importante que se visualice correctamente para mostrar al usuario lo que este capta.

Modificar los parámetros de la cámara: Con esta orden, *Camera.Parameters parameters = mCamera.getParameters()* se pueden acceder y configurar multitud de parámetros [59], como por ejemplo el modo del flash, el enfoque, el tamaño de la foto, el tamaño del preview, el balance de blancos, etc. No todos los dispositivos soportan todos los parámetros que se indica en [55], por lo que si se va a usar alguno muy especifico es necesario que se compruebe si se soporta.

Como se trata de hacer una aplicación que use la cámara desde cero, es importante configurar estos parámetros para que se adecuen a la aplicación. En este caso es importante configurar bien el flash de la cámara para poder eliminar sombras que puedan molestar para la segmentación y el autofoco para enfocar mejor la escena. Se elegirá un tamaño de foto de 3MPx, que es suficiente para tener una buena calidad de imagen.

Fijar la orientación del preview: *Camera.setCameraDisplayOrientation()* podemos cambiar la orientación que queramos que tenga el preview. Es un paso importante si queremos que nuestra aplicación funcione sólo en modo portrait ya que si no no se mostraran las imágenes correctamente por pantalla. Esta aplicación se mostrará sólo en este modo.

Hacer una foto: Se usa el método *Camera.takePicture()* para tomar una foto una vez que el preview ha empezado.

Reiniciar el preview: Una vez que se ha tomado foto, el preview parará automáticamente y hay que volver a llamar al método *Camera.startPreview()* para volver a enviar imágenes a la pantalla. No hay un lugar específico en el código en el que se tenga que reiniciar el preview, depende del uso de la aplicación.

Parar el preview y liberar el objeto Camera: Cuando la aplicación ya ha acabado de usar la cámara, es importante liberarla llamando la método *Camera.release()* para que no haya problemas con otras aplicaciones que quieran hacer uso de este recurso.

Se recomienda que antes de liberar o incluso antes de llamar al método *Camera.startPreview()* se pare la cámara con el método Camera.stopPreview() para evitar conflictos.

Las siluetas que se mostraran por pantalla a la vez que lo que capta la cámara son las siguientes:

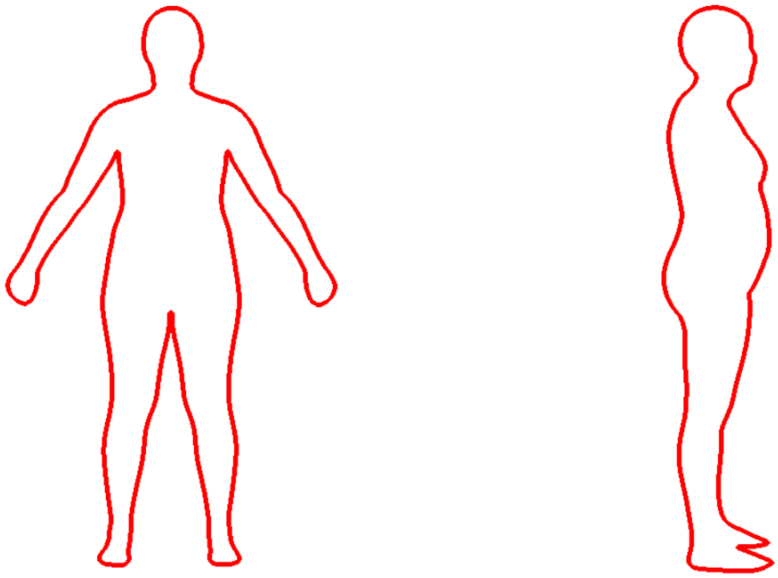


Figura 28. Figuras que se muestran por la pantalla del dispositivo para indicar la posición que tiene que adoptar el usuario. Ambas son transparentes. Fuente: IBV

Las siluetas son de un cuerpo femenino ya que la base de datos 3D que se utilizará para la reconstrucción será la del estudio antropométrico de la población femenina en España realizado por el IBV, por lo que en principio la aplicación estaría dirigida a mujeres.

Las fotos se realizan tocando en cualquier parte la pantalla del teléfono. Después de cada foto se mostrará un mensaje de espera, tras el cual se mostrará el resultado de la segmentación, dándole al usuario la posibilidad de volver a repetir la captura si no se obtiene un buen resultado o continuar adelante en la aplicación.

4.3. Sensores

Una función que va a necesitar esta aplicación es obtener la inclinación del teléfono y que nos ayudara a la hora de corregir determinados errores que tienen que ver con la posición a la hora de tomar las fotos y que pueden hacer que no salgan bien los perfiles a obtener y que no se reconstruya el perfil 3D. Para conseguir esto nos vamos a basar en el manejo de los sensores que nos proporciona Android.

Android nos permite acceder a muchos tipos de sensores [42], algunos basados en hardware que están directamente instalados en el dispositivo y que sus datos son obtenidos midiendo directamente propiedades ambientales específicas como puede ser el campo geomagnético o el cambio angular y otros basados en software que obtienen sus datos a partir de uno o más de los primeros, como por ejemplo el acelerómetro lineal o el sensor de gravedad.

Lo normal es que un dispositivo no tenga todos los sensores instalados. La mayoría incluye acelerómetro y magnetómetro, incluso puede ser que algún dispositivo tenga varios de un tipo de sensor, por ejemplo que tenga dos sensores de gravedad con distinto rango de medida.

Los sensores soportados por la plataforma Android se pueden dividir en:

Sensores de movimiento. Miden fuerzas de aceleración y rotación en los tres ejes.

Sensores ambientales. Miden por ejemplo la temperatura, la humedad o la iluminación.

Sensores de posición. Miden la posición del dispositivo.

Sensor	Tipo	Descripción	Usos comunes
TYPE_ACELEROMETER	Hardware	Mide la fuerza de la aceleración que se aplica al dispositivo en los tres ejes, incluyendo la fuerza de la gravedad	Detección de movimiento (tilt, agitación, etc)
TYPE_AMBIENT_TEMPERATURE	Hardware	Mide la temperatura ambiente en grados centígrados	Monitorizar la temperatura
TYPE_GRAVITY	Software o Hardware	Mide la fuerza de la gravedad aplicada al dispositivo en los ejes físicos	Detección de movimiento (tilt, agitación, etc)
TYPE_GYROSCOPE	Hardware	Mide el ratio de rotación del dispositivo alrededor de los tres ejes físicos	Detección de la rotación
TYPE_LIGHT	Hardware	Mide la luz ambiental	Controlar el brillo de la pantalla
TYPE_LINEAR_ACCELERATION	Software o Hardware	Mide la fuerza de la aceleración que se aplica al dispositivo en los tres ejes son contar la fuerza de la gravedad	Monitorizar la aceleración en cada eje
TYPE_MAGNETIC_FIELD	Hardware	Mide el campo magnético para los tres ejes físicos	Crear una brujula
TYPE_ORIENTATION	Software	Mide los grados de rotación de un dispositivo alrededor de los tres ejes físicos. Desde el API 3 esta función no se ha continuado.	Determinar la posición del dispositivo
TYPE_PRESURE	Hardware	Medir la presión del aire	Monitorizar cambios de presión en el aire
TYPE_PROXIMITY	Hardware	Mide la proximidad de un objeto en relación con la pantalla del dispositivo	Determinar la posición del teléfono durante una llamada
TYPE_RELATIVE_HUMIDITY	Hardware	Mide la humedad ambiental	
TYPE_ROTATION_VECTOR	Software o Hardware	Mide la orientación de un dispositivo proporcionando los tres elementos del vector de giro del dispositivo.	Detección de movimiento y rotación
TYPE_TEMPERATURE	Hardware	Mide la temperatura del dispositivo en grados centígrados.	Monitorizar temperaturas

Tabla 1. Diferentes sensores soportados por Android. Fuente: [42]

Para poder acceder a los sensores disponibles en un dispositivo, Android nos proporciona las siguientes clases e interfaces que son parte del paquete android.hardware:

SensorManager. Esta clase nos permite usar una instancia del servicio del sensor. Nos proporciona varios métodos para listar y acceder a los sensores y registrar eventos.

Sensor. Esta clase nos permite crear una instancia de un sensor específico. Ofrece varios métodos para determinar las capacidades del sensor.

SensorEvent. Ofrece información sobre un evento de un sensor, como pueden ser datos en bruto del sensor o el tipo de sensor que ha provocado el evento.

SensorEventListener. Interface con dos métodos callback que reciben notificaciones cuando por ejemplo cambian los datos del sensor.

En una aplicación que requiera el uso de sensores se tienen que seguir dos tareas básicas:

Identificar el/los sensor/es y sus capacidades.

Para identificar los sensores disponibles en el dispositivo primero se necesita obtener una referencia a sensor service. Se tiene que crear una instancia de la clase SensonManager y llamar al método getSystemService() pasándole como argumento SENSOR_SERVICE.

```
Private SensorManager mSensorManager;

…

mSensorManager = (SensorManager)getSystemService(Context.SENSOR_SERVICE);
```

Luego se pueden listar todos los sensores del dispositivo con *getSensorList()* pasandole como parámetro TYPE_ALL.

```
List<Sensor> deviceSensors = mSensorManager.getSensorList(Sensor.TYPE_ALL);
```

O si se quiere listar unos determinados tipos de sensores, se puede sustituir TYPE_ALL por TYPE_GRAVITY, TYPE_GYROSCOPE, etc.

También se puede comprobar si un determinado tipo de sensor existe en el dispositivo con el método getDefaultSensor() pasándole como parámetro un tipo de sensor.

```
Prívate SensorManager mSensorManager;

…

mSensorManager = (SensorManager)getSystemService(Context.SENSOR_SERVICE);

if(mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD!=null)){

        //Acierto. Hay un magnetómetro.

}else{

        //Fallo. No hay un magnetómetro.

}
```

Monitorizar los eventos del Sensor.

Se lleva a cabo con la implementación de dos métodos callback:

1. Cuando ocurre un cambio en la precisión del sensor el sistema llama a onAccuracyChanged(), devolviendo una refetencia al objeto sensor que ha cambia y la nueva precisión.

2. Cuando el sensor devuelve un nuevo valor medido, el sistema llama al método onSensorChanged(), devolviendo un objeto del tipo SensorEvent que contiene información de los datos capturados del sensor.

Hay más información y usos sobre sensores en la web de desarrolladores de Android, pero para este proyecto con el uso básico escrito anteriormente es suficiente. Para más información se recomienda visitar la página web para desarrolladores de Android en su apartado de sensores [42].

4.3.1. Sensor de gravedad

Para calcular la inclinación del teléfono a la hora de hacer las fotos, para que sirva como calibración de la cámara a la hora de recontruir el modelo 3D, se va a hacer uso del sensor de gravedad del dispositivo. Necesitaremos obtener los valores que se obtienen en cada uno de los ejes para luego poder calcular la matriz de rotación, aunque sólo hará falta conocer los valores de los ejes x y z, ya que sólo con estos dos ejes se obtienen los valores necesarios.

Es importante conocer el sistema de coordenadas que se usa en los sensores. Este puede verse en la figura 29.

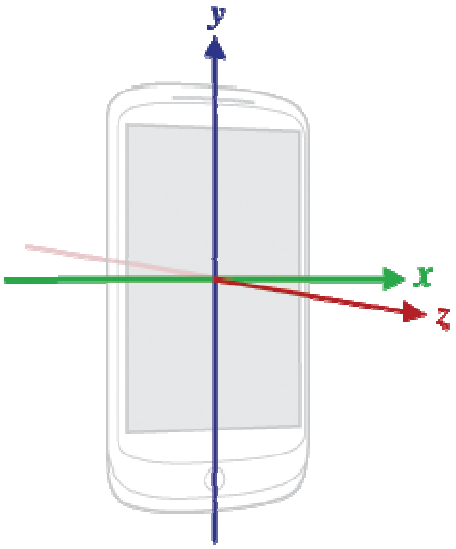


Figura 29. Sistema de coordenadas usado por los sensores. Fuente: [42]

Para monitorizar los valores devueltos por este sensor lo único que hay que hacer es llamar al método onSensorChanged() y almacenar los valores devueltos por cada coordenada:

```
public void onSensorChanged(SensorEvent event) {

        gx = event.values[0];

        gy = event.values[1];

        gz = event.values[2];

}
```

Una vez obtenidos los valores del sensor de gravedad se calcula la matriz de rotación, que tendrá la siguiente forma:

$$\begin{bmatrix} 1 & -gx/g & 0 \\ gx/g & 1 & -gz/g \\ 0 & gz/g & 1 \end{bmatrix}$$

Siendo g el valor de la gravedad obtenido mediante la raíz de la suma de cada componente al cuadrado ($g = \sqrt{gx^2 + gy^2 + gz^2}$).

4.3. Librería OpenCV

Lo primero que hay que hacer es cargar la librería de OpenCV al principio de la aplicación, ya que esta tarda un poco en cargar y si queremos usar enseguida alguna de las funciones que ofrece nos saldría un error. Se carga en el método onCreate en la clase MainActivity, tal y como se puede comprobar en el anexo B.

En los siguientes apartados se explicara el código empleado para el funcionamiento de los dos métodos para tratamiento de imagen explicados en el apartado 3.

4.3.1. Algoritmo watershed

Para el uso de este algoritmo, OpenCV tiene la función Imgproc.watershed() [50] para su uso en Java, con los siguientes parámetros de entrada:

- image: imagen de entrada en formato 8-bit 3-channel. Es la fotografía tomada por la cámara del teléfono sin modificar.
- markers: almacena tanto los marcadores empleados para la segmentación como el resultado de esta. Es de tipo 32-bit single-channel.

Ahora veamos cómo se preparan todos los datos para que la función haga la segmentación correctamente. Primero se cargará una de las siluetas en función de si estamos segmentando una foto con el usuario de frente o de perfil (son las mismas que se muestran por pantalla (figura 28) pero preparadas para generar los marcadores) y su correspondiente marcador para indicar lo que será foreground.

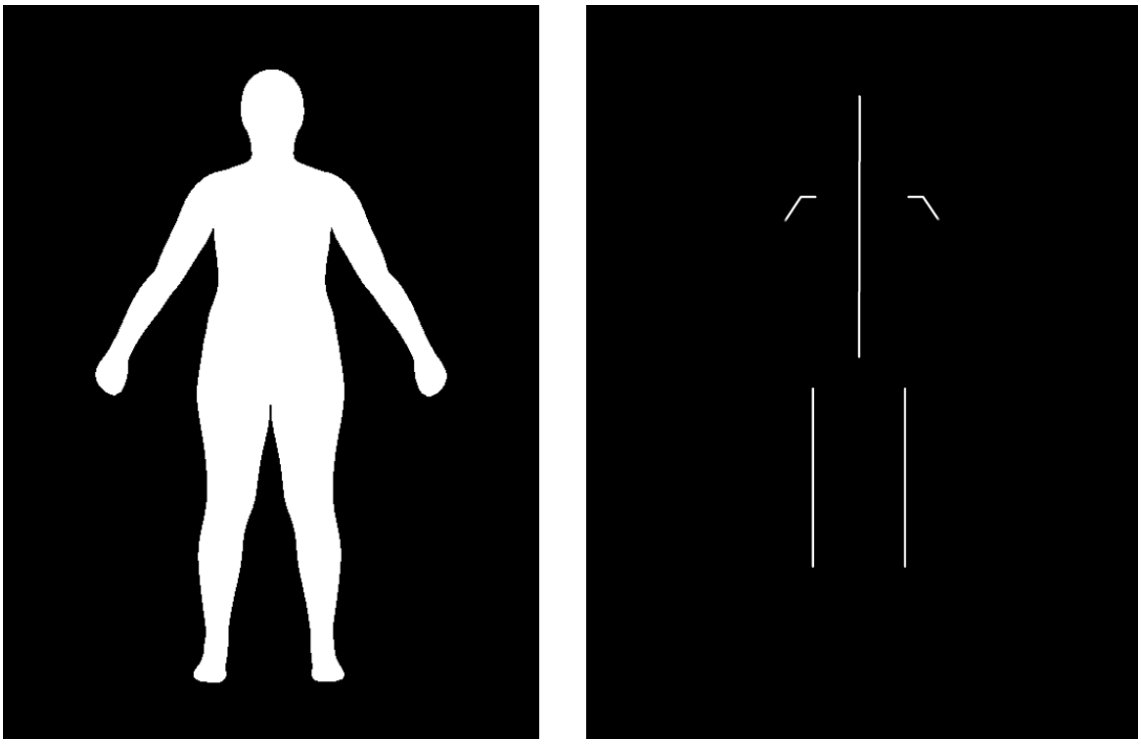


Figura 30. Figuras iniciales de frente para crear el marcador final de frente. Derecha: indica lo que será foreground

Se dilatará la figura inicial con la función de OpenCV Imgproc.dilate() [50] para eliminar ruido y abarcar más espacio para que si la persona no sale del todo centrada en la silueta, aún se pueda hacer la segmentación.

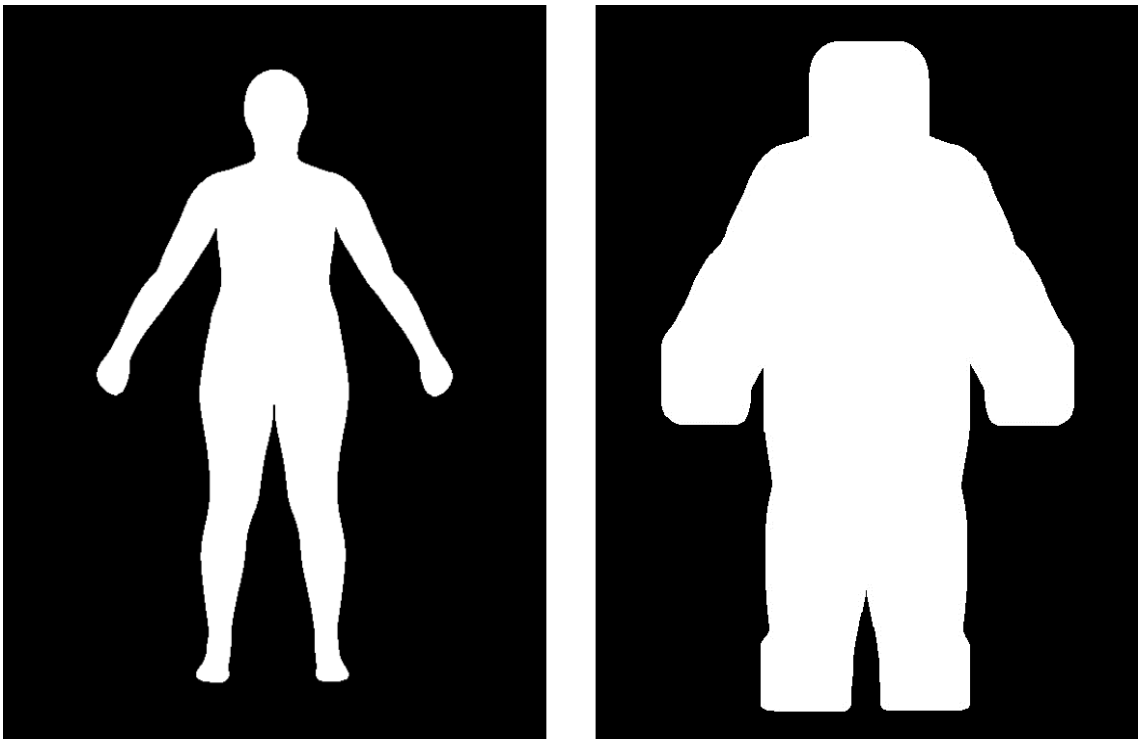


Figura 31. Dilatación empleada para la creación del marcador de frente

Todas las imágenes usadas tienen que tener el mismo tamaño, ya que si no el algoritmo no segmentará bien, aparte de que al llamar a la función watershed de opencv saldrá un error. Las fotografías del usuario de toman con una resolución de 3 Megapíxel (2048x1536) para obtener una buena resolución para poder distinguir bien los bordes de las figuras y que no suponga mucha carga a la CPU del teléfono para manejarla.

Pero esta resolución en las fotografías es demasiado grande para estos algoritmos, ya que tardan mucho en hacer los cálculos e incluso a veces suelen dar errores. Por eso se baja la resolución. La altura se deja fija a 750 pixeles y la anchura se calcula en función de la altura y anchura de la imagen capturada, mediante la siguiente relación: newW=(oldW/oldH)*newH, siendo oldW y oldH la anchura y altura respectivamente de la foto original y newW y newH las de la foto destino.

El marcador usado para segmentación puede verse en la figura 32. Lo que se marca como 0 indica que es una región desconocida y que tendrá que ser el algoritmo el que decida si es background o foreground. Lo marcado como 1 indica que es seguro foreground y el resto se marca como 128, indicando que esa zona no será tomada en cuenta.

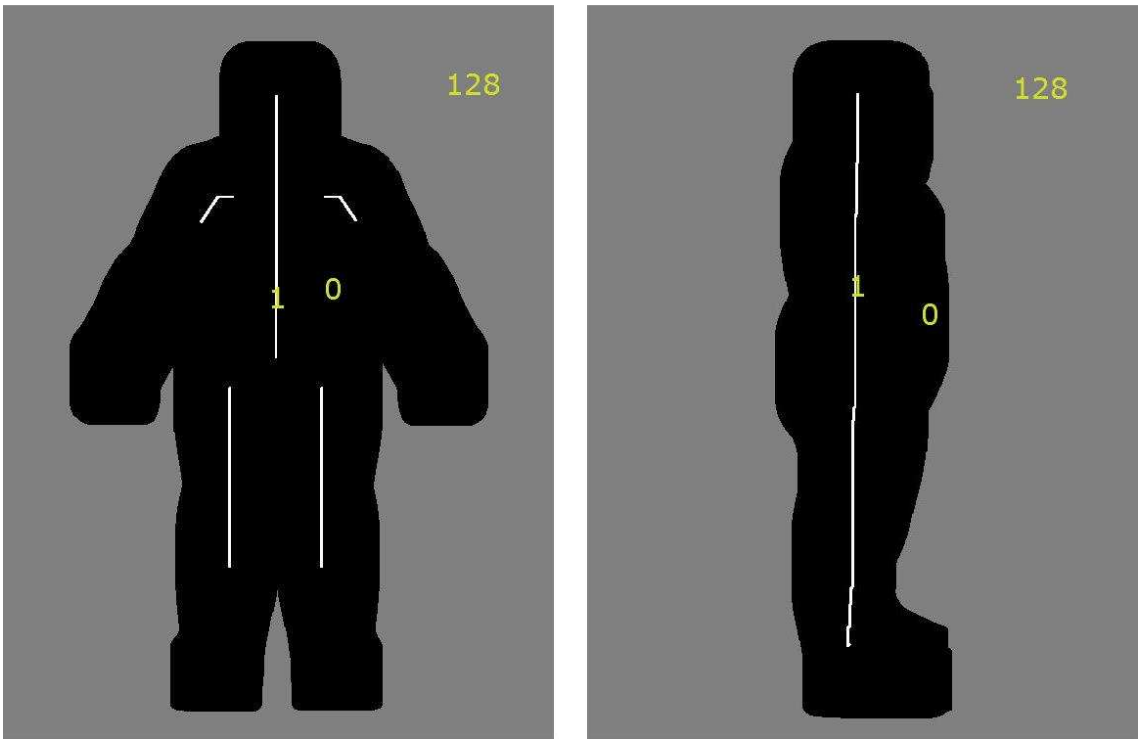


Figura 32. Marcadores usados de frente y de perfil para el algoritmo watershed

Una vez tenemos todos los datos preparados ya podemos llamar a la función Imgproc.watershed() y con el resultado almacenado en markers buscamos los contornos con la función de OpenCV Imgproc.findContours() que luego dibujaremos con otra función de OpenCV, Imgproc.drawContours(), [50], sobre la imagen capturada con la cámara para así poder mostrar al usuario en la pantalla de su teléfono móvil el resultado de la segmentación y que pueda comprobar si el resultado es correcto o hay que volver a repetirlo.

4.3.2. Algoritmo grabcut

Para el uso de este algoritmo OpenCV tiene la función Imgproc.grabCut() [50] para su uso en Java, con los siguientes argumentos:

- img: imagen de entrada. Tiene que ser una imagen con tres canales. Es la fotografía tomada por la cámara del teléfono sin modificar.
- mask: es una imagen en la que se especifica que áreas serán background, foreground o probable background/foreground. Se consigue marcando estas zonas como 0, 1, 2 o 3. Imagen de un solo canal. También se almacena la imagen resultado de la segmentación.
- rect: indica las coordenadas del rectángulo marcado por el usuario, pero en este caso no se le pasa nada ya que el usuario no interviene a la hora de hacer la segmentación.
- bdgModel, fgdModel: arrays usados internamente por el algoritmo. En este caso no cumplen ninguna función y se pasan vacios.
- iterCount: número de iteraciones.
- mode: se puede indicar si se va a usar una máscara o un rectángulo para hacer la segmentación. En este caso se indicará que se usa una máscara.

Ahora veamos cómo se preparan todos los datos para que la función haga la segmentación correctamente. La primera parte es igual que en el anterior algoritmo. Primero se cargará una de las siluetas en función de si estamos segmentando una foto con el usuario de frente o de perfil y su correspondiente marcador para indicar lo que será el foreground.

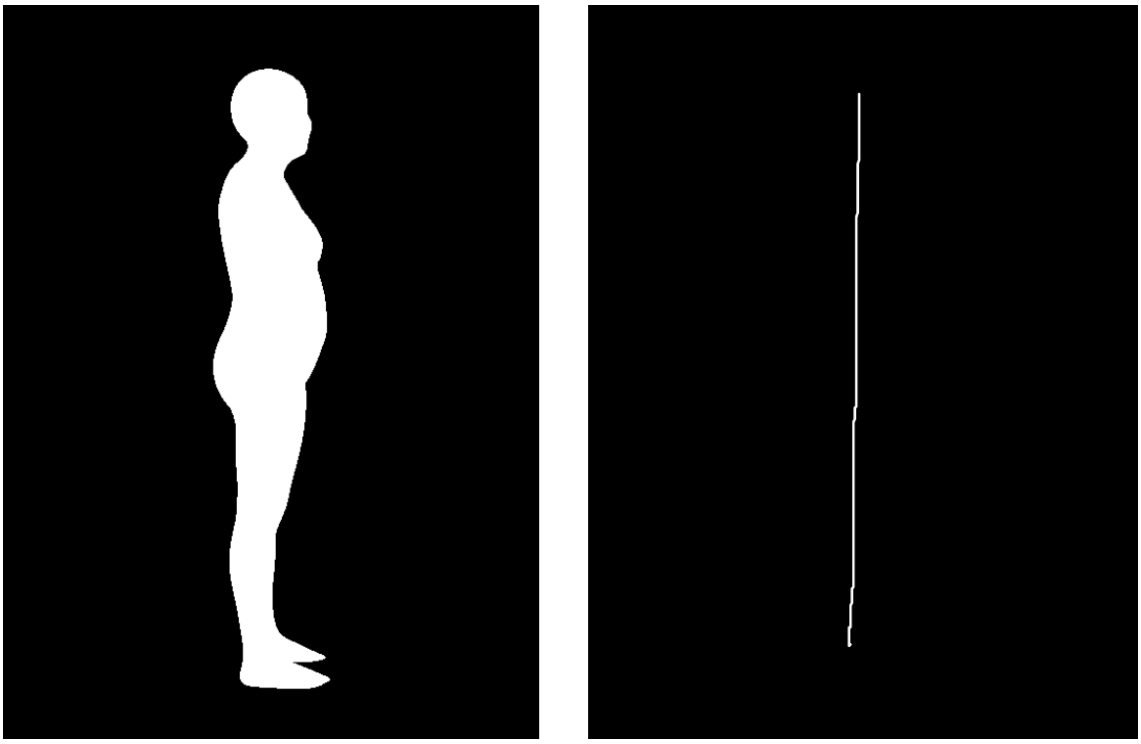


Figura 33. Izquierda: Figura inicial de perfil para crear el marcador final. Derecha: Indica lo que será foreground

Al igual que antes se dilata la figura inicial.

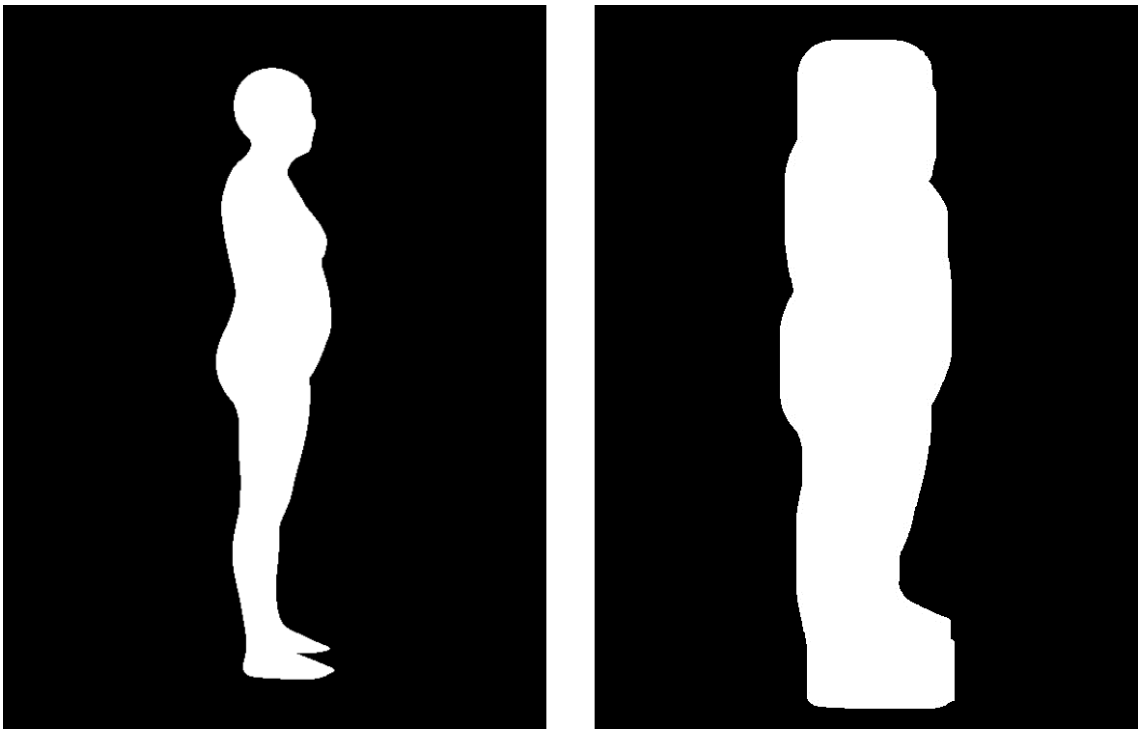


Figura 34. Dilatación empleada para la creación del marcador de perfil

Al igual que antes se reduce el tamaño de las imágenes empleando la misma fórmula.

El punto crítico es montar la máscara. Vamos a necesitar una máscara que tenga marcadas las zonas como 0, 1, 2 o 3 para indicar si la zona es background, foreground o probable background/foreground respectivamente. Esta puede verse en la siguiente figura.

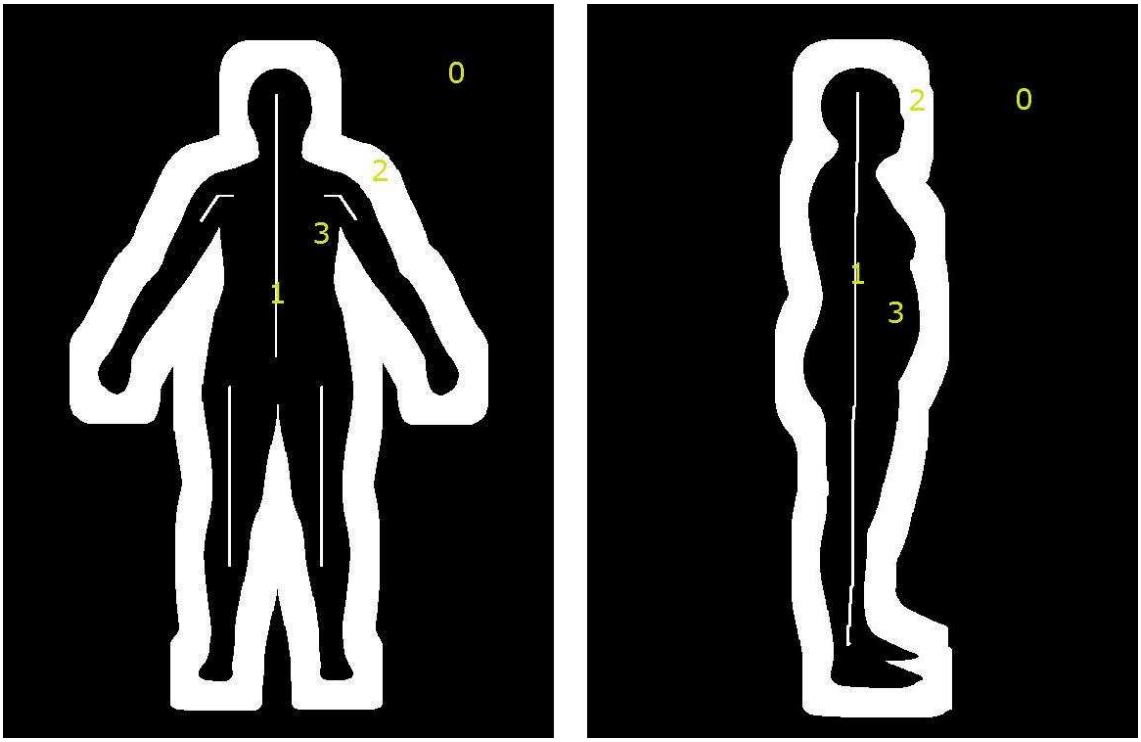


Figura 35. Marcadores usados de frente y perfil para el algoritmo grabcut

Una vez tenemos todos los datos preparados ya podemos llamar a la función Imgproc.grabCut() y con el resultado almacenado en mask buscamos los contornos con la función de OpenCV Imgproc.findContours() que luego dibujaremos con otra función de OpenCV, Imgproc.drawContours(), [50], sobre la imagen capturada con la cámara para así poder mostrar al usuario en la pantalla de su teléfono móvil el resultado de la segmentación y que pueda comprobar si el resultado es correcto o hay que volver a repetirlo.

En ambos casos mientras se realizan los cálculos se muestra un mensaje de espera mediante un ProgressDialog, que no es más que una rueda de espera en la que se puede introducir más información para mostrar por pantalla [42]. Todos estos cálculos se realizan en segundo plano con la clase AsynTask mientras se muestra el mensaje por pantalla, y al acabar desaparece el mensaje de espera y se muestra por pantalla el resultado de la segmentación. El código se puede ver en la clase CameraMain en el anexo B. El esqueleto de la clase AsyncTask se muestra a continuación (no tienen porque usarse todos los métodos de esta clase, sólo los que necesitemos).

```
private class MiTareaAsincrona extends AsyncTask<Void, Integer, Boolean> {

   @Override
   protected void onPreExecute() {
         //Se ejecutará antes del código principal de nuestra tarea
   }

   @Override
   protected Boolean doInBackground(Void... params) {
         //Contendrá el código principal de nuestra tarea.
   }

   @Override
   protected void onProgressUpdate(Integer... values) {
         //Se invoca cada vez que llamamos al método publishProgress() desde el método
doInBackground().
   }

   @Override
   protected void onPostExecute(Boolean result) {
         //Se ejecutará cuando finalice nuestra tarea.
   }

   @Override
   protected void onCancelled() {
         //Se invocará cuando cancelemos nuestra tarea.
   }
}
```

Figura 36. Izquierda: Mensaje de espera de resultados. Derecha: Mensaje de confirmación de resultados

4.4. Almacenamiento de resultados.

El fichero que se almacena en la memoria del teléfono y que posteriormente se enviará al servidor dedicado es un fichero en formato XML que ocupa entre 10 y 15 KB, dependiendo sobre todo del tamaño del resultado de la segmentación.

En este fichero se tiene que almacenar:

- Altura, peso y edad del usuario.
- El tamaño de la fotografía empledada para la segmentación (con la altura fijada a 750 pixeles).
- La distancia focal de la cámara del dispositivo empleado. Se obtiene con la función [59]:

```
Camera.Parameters parameters=mCamera.getParameters();
focalLength =parameters.getFocalLength();
```

- El ángulo vertical y horizontal de la cámara del dispositivo. Se obtiene al igual que la distancia focal del objeto Camera.parameters.

```
verticalAngle =parameters.getVerticalViewAngle();
horizontalAngle_frente=parameters. getHorizontalViewAngle();
```

- La matriz de rotación y los datos capturados por el sensor para saber la inclinación del teléfono tanto de la fotografía de frente como de la de perfil.
- Los contornos de la segmentación tanto de la fotografía tanto de frente como de la de perfil almacenados en una matriz de dos columnas y varias filas en función del tamaño de la segmentación.

Capítulo 5: Pruebas y Resultados

En este apartado se van a comparar los resultados obtenidos por los dos métodos con distintas personas y se elegirá el que mejor resultados dé para la aplicación final. Es complicado que el usuario encaje del todo bien en la silueta a la primera ya que este no puede verse y el que hace las fotos tiene que ir dando instrucciones para que se ajuste a la silueta. Pero eso no supondrá un problema a la hora de segmentar tal y como veremos más adelante.

Las pruebas se han realizado en un Samsung Galaxy Ace 2 [51], modelo número GT-I8160P con versión de Android 2.3.6 y 512 megas de RAM para probar los dos métodos de tratamiento de imagen elegidos.

Las pruebas se realizaron en un laboratorio en el Instituto de Biomecánica de Valencia (IBV). Para las medidas se uso ropa oscura y ajustada, para que la ropa no genere pliegues con los que se deformaría la silueta del usuario y no se reconstruiría correctamente la silueta 3D, y con fondo blanco, ya que el contraste entre la ropa oscura y este color de fondo ayudan a la hora de realizar la segmentación.

Como la primera foto que se tomará en la aplicación es la de frente, empezaremos comentando primero estos resultados. La figura 37 derecha muestra cómo se vería la pantalla del teléfono antes de tomar la foto con el usuario y la silueta que sirve para marcar la posición en la que se tiene que colocar. Como es lógico esta parte es la misma tanto para watershed como para grabcut.

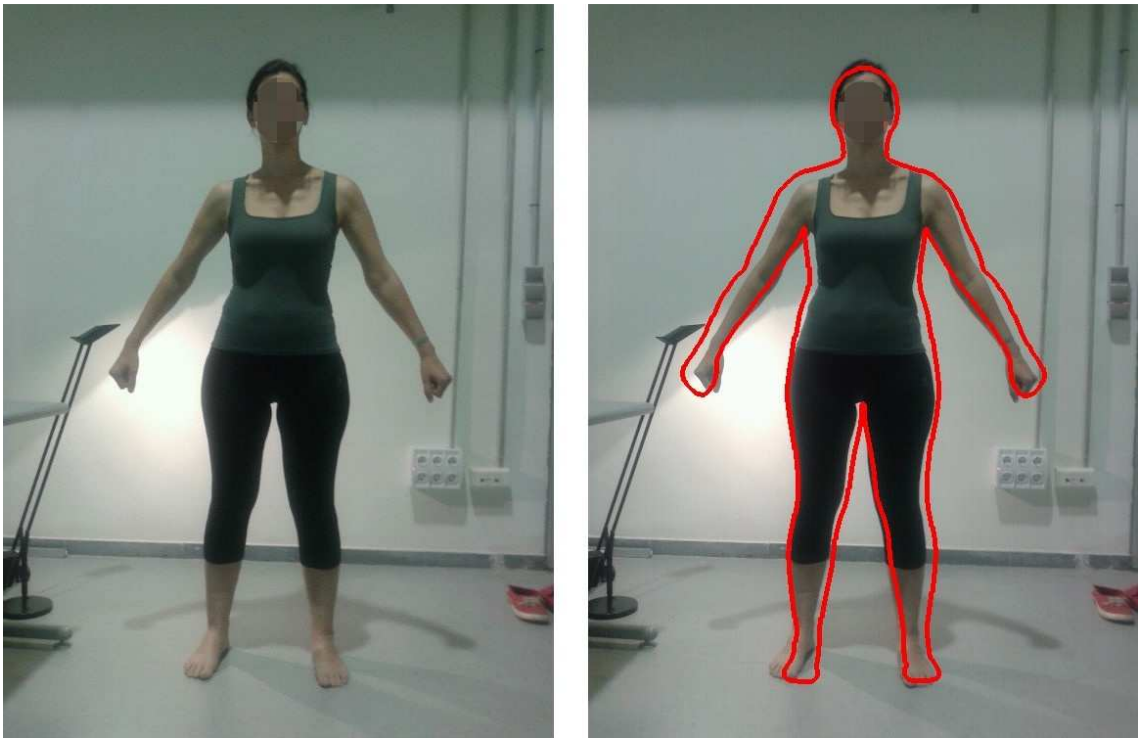


Figura 37. Imagen modelo A e imagen modelo A con silueta de frente

En la siguiente figura vemos como los marcadores que sirven para indicar lo que sería seguro foreground tocan la zona que queremos segmentar. Esta foto sería igual tanto para watershed como para grabcut.

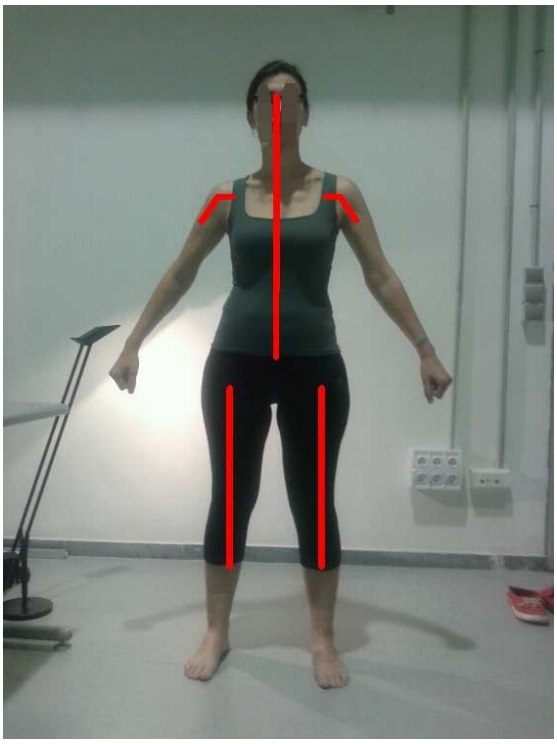


Figura 38. Fotografía modelo A con los marcadores de frente

En la siguiente figura podemos ver ya el resultado de los métodos. El de la izquierda sería el resultado devuelto por watershed y el de la derecha el devuelto por grabcut. Para poder apreciar mejor el resultado y poder mostrarlo por la pantalla del dispositivo, desde estas fotos se calcularía el contorno de las figuras y se dibujarían sobre la foto original, tal y como puede apreciarse en la figura 40.

Figura 39. Resultado de la segmentación. Izquierda watershed. Derecha grabcut

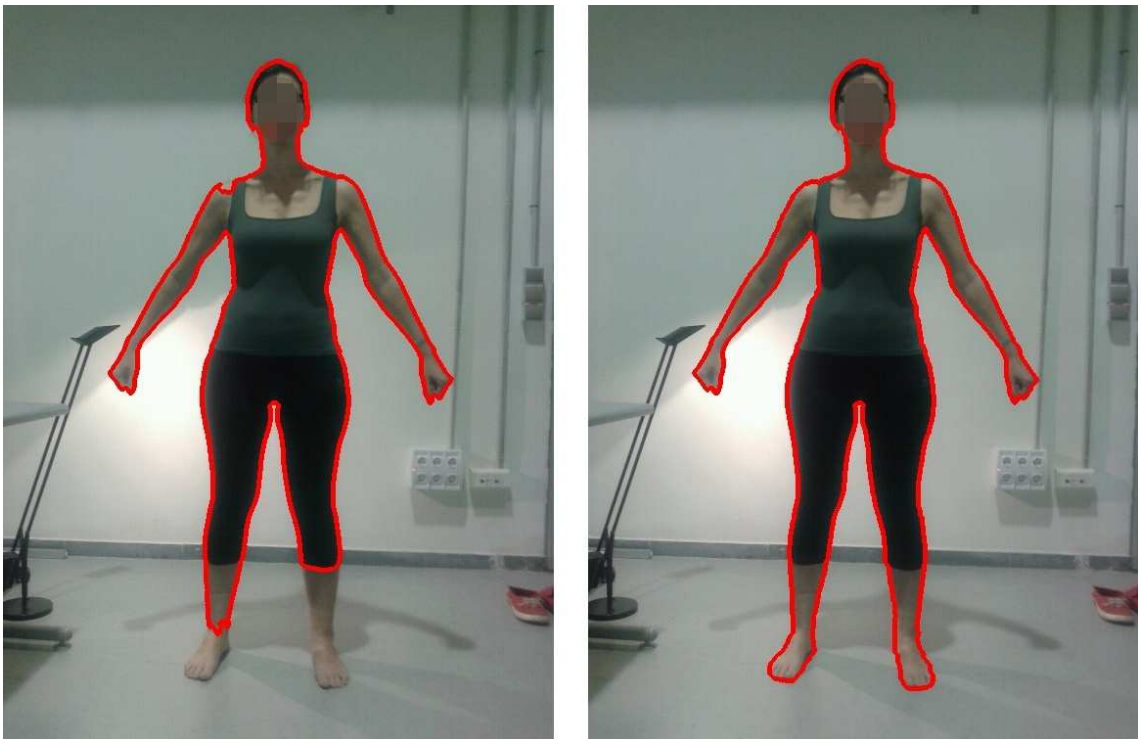


Figura 40. Resultado de la segmentación sobre la foto original de frente. Izquierda: watershed. Derecha: grabcut

Es bastante evidente que el resultado obtenido en grabcut es mucho más preciso a pesar de que el usuario no encajaba del todo bien en la silueta, lo cual es complicado ya que la silueta se ha obtenido como promedio de varias. Al usar marcadores no muy gruesos para indicar la zona a segmentar y no hacerlos llegar hasta el final de las extremidades (tanto brazos como piernas) y como la zona alrededor de la silueta se marca como probable background (tal y como se vió en el apartado 4.3), se le da al usuario un cierto grado de error ya que al hacerse las fotos con un dispositivo que por lo general sólo estará sujeto con los brazos de la persona que realiza la foto siempre puede moverse un poco. El problema con watershed es que la zona indicada como foreground no es lo suficientemente grande y quedan muchos huecos entre los marcadores y no acaba de segmentar del todo bien.

Con las fotos de perfil se procede de igual manera. En la figura siguiente se muestra cómo se vería en la pantalla de un teléfono (imagen de la derecha).

Figura 41. Imagen original e imagen original con marcador de perfil

Como se puede ver en la siguiente figura el marcador sólo toca la zona que queremos segmentar. Es el mismo para watershed y grabcut. El marcador no va del todo derecho desde la cabeza hasta los pies porque si no se saldría de la zona a segmentar y tocaría el fondo y del resultado de la segmentación sería erróneo.

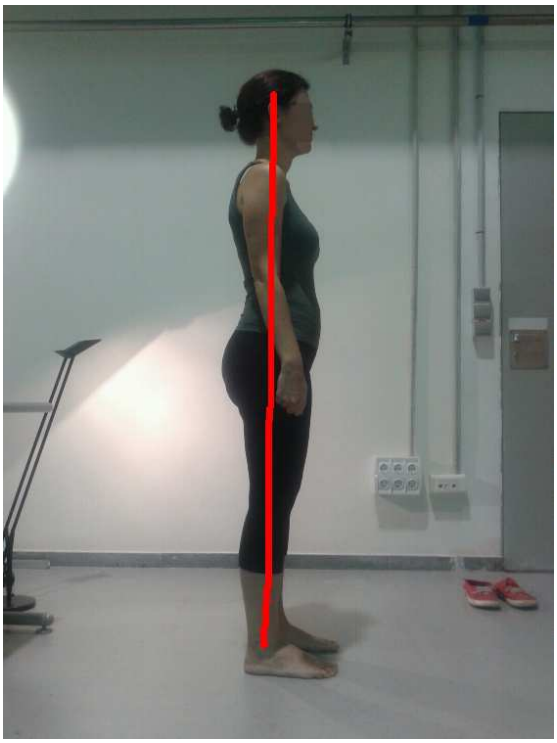


Figura 42. Imagen original con el marcador de perfil

En la figura siguiente, la fotografía de la izquierda se corresponde con el resultado de la segmentación usando watershed y la fotografía de la derecha con la de grabcut.

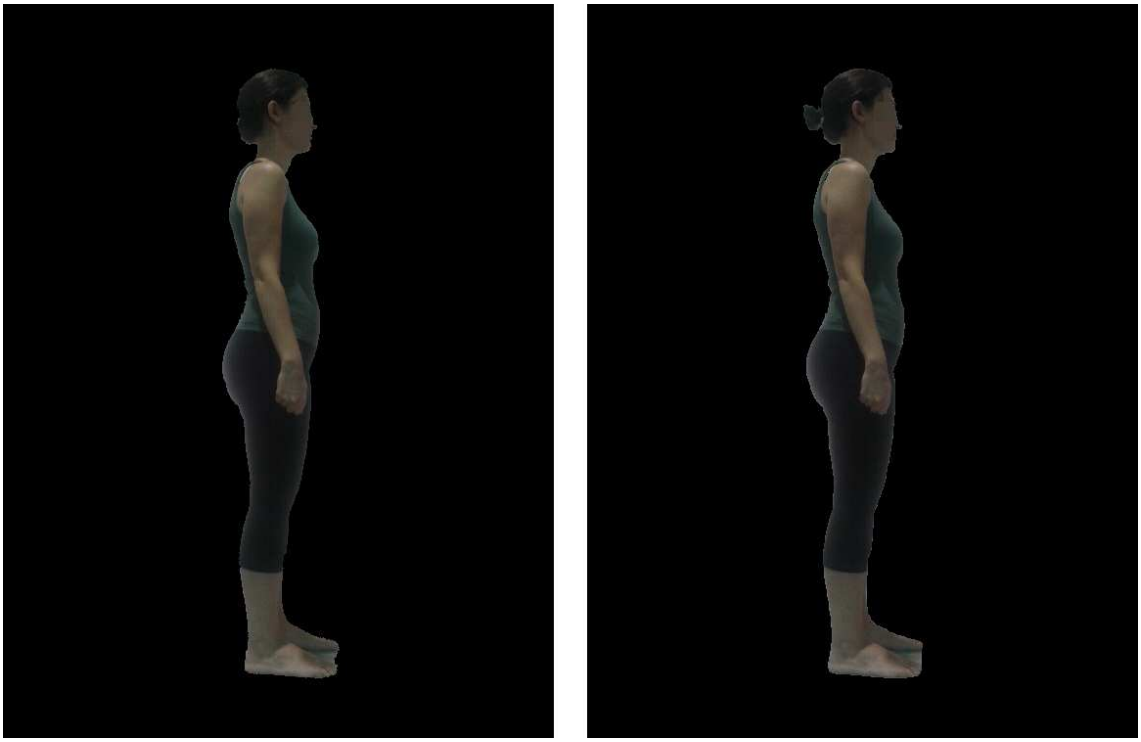


Figura 43. Resultado de la segmentación. Izquierda: watershed. Derecha: grabcut

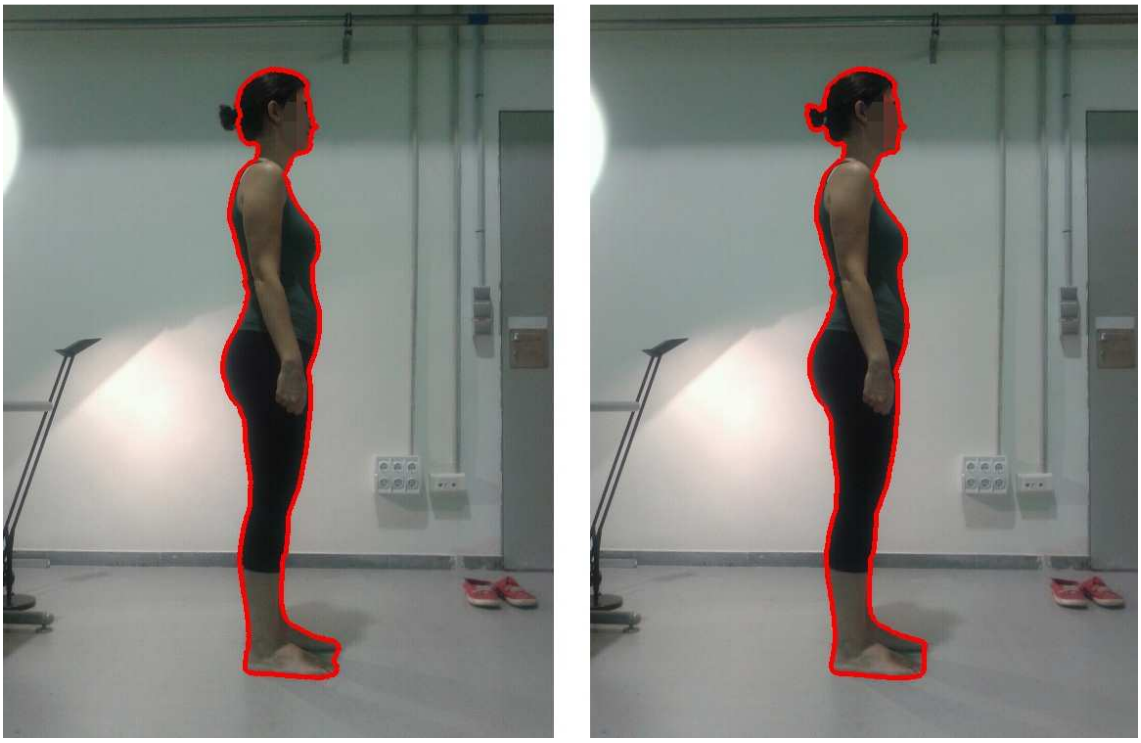


Figura 44. Contorno de la segmentación dibujado sobre la fotografía original de perfil. Izquierda: watershed. Derecha: grabcut

En este caso puede verse que el resultado es muy parecido, la zona a segmentar es más lisa, brazos y piernas están pegados al cuerpo y no separados como en la fotografía de frente y esto

hace que los dos algoritmos se comportan más o menos igual. A continuación se muestran otros ejemplos del uso de estos métodos de segmentación.

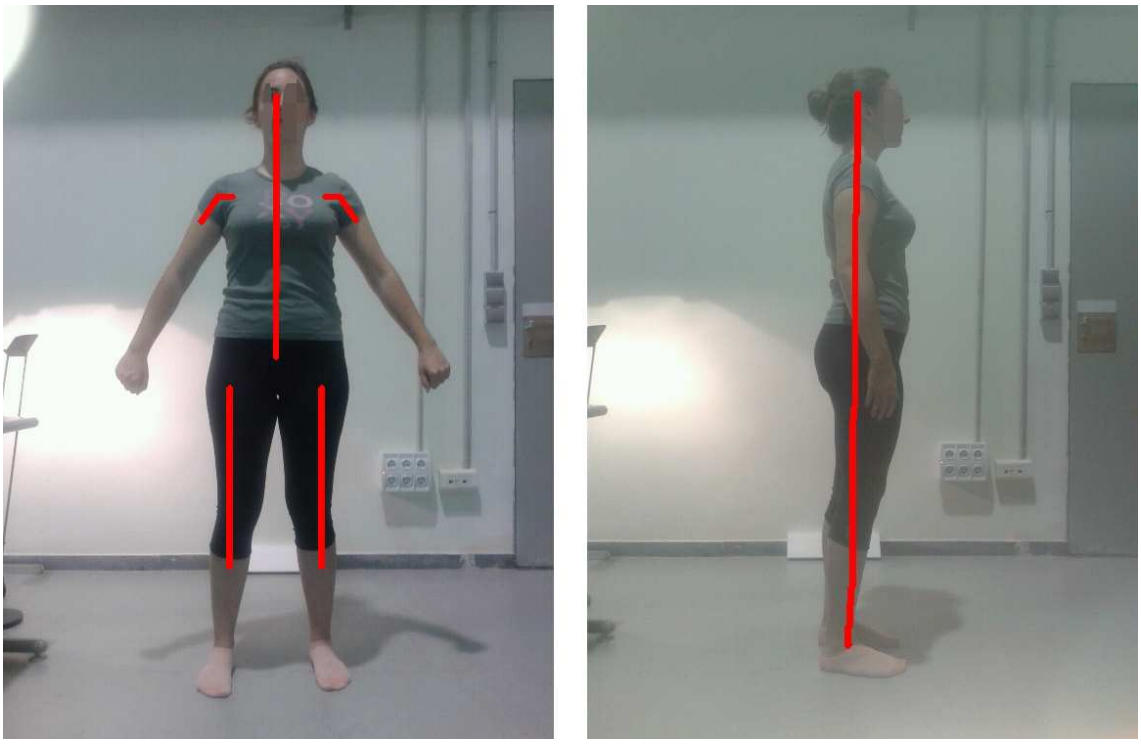


Figura 45. Marcadores para foreground sobre modelo B

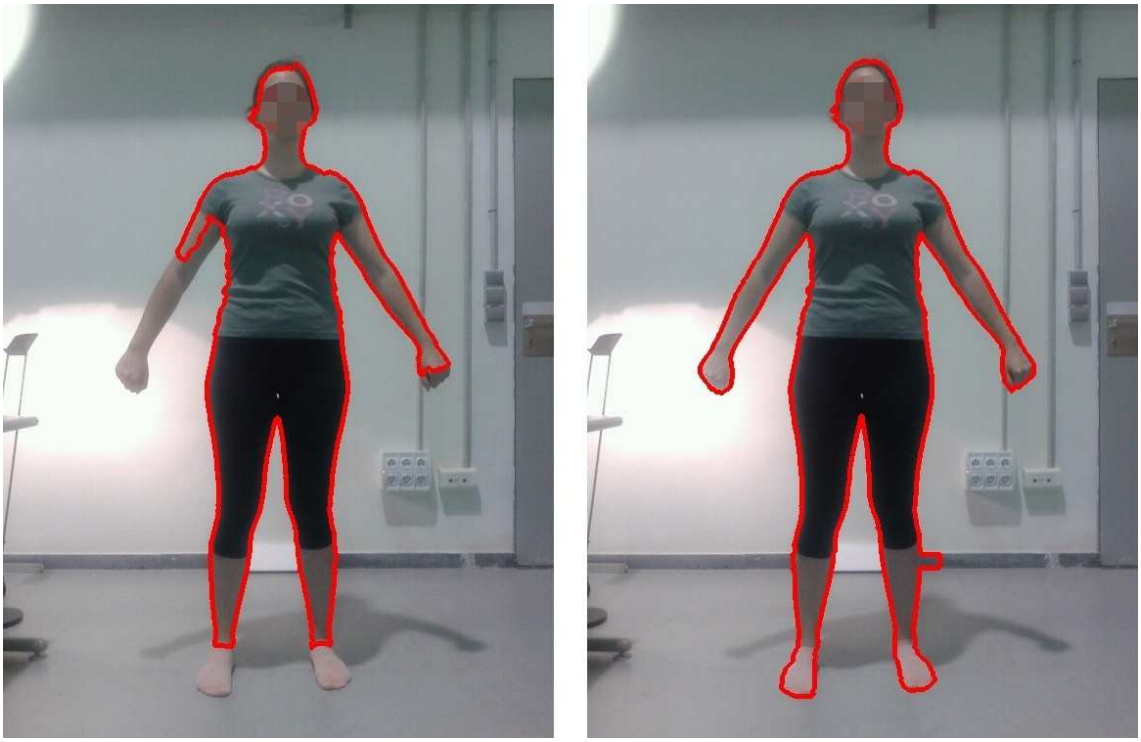


Figura 46. Ejemplo de segmentación modelo B posición frontal. Izquierda: watershed. Derecha: grabcut

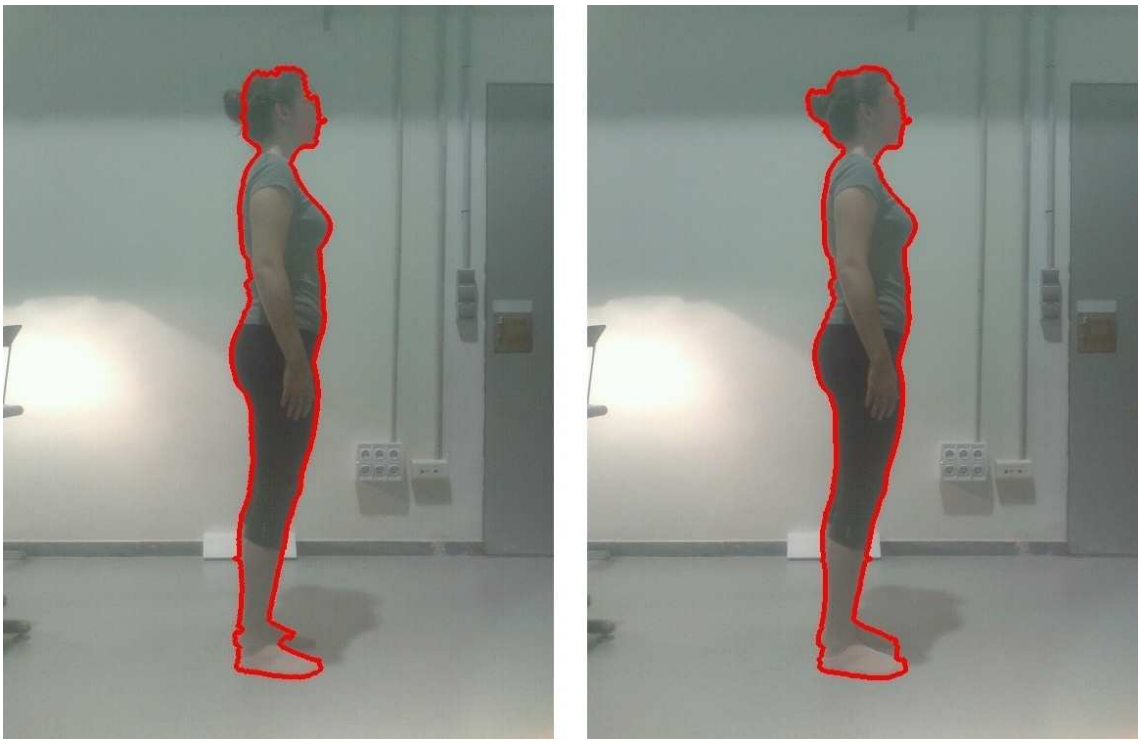


Figura 47. Ejemplo de segmentación modelo B posición lateral. Izquierda: watershed. Derecha: grabcut

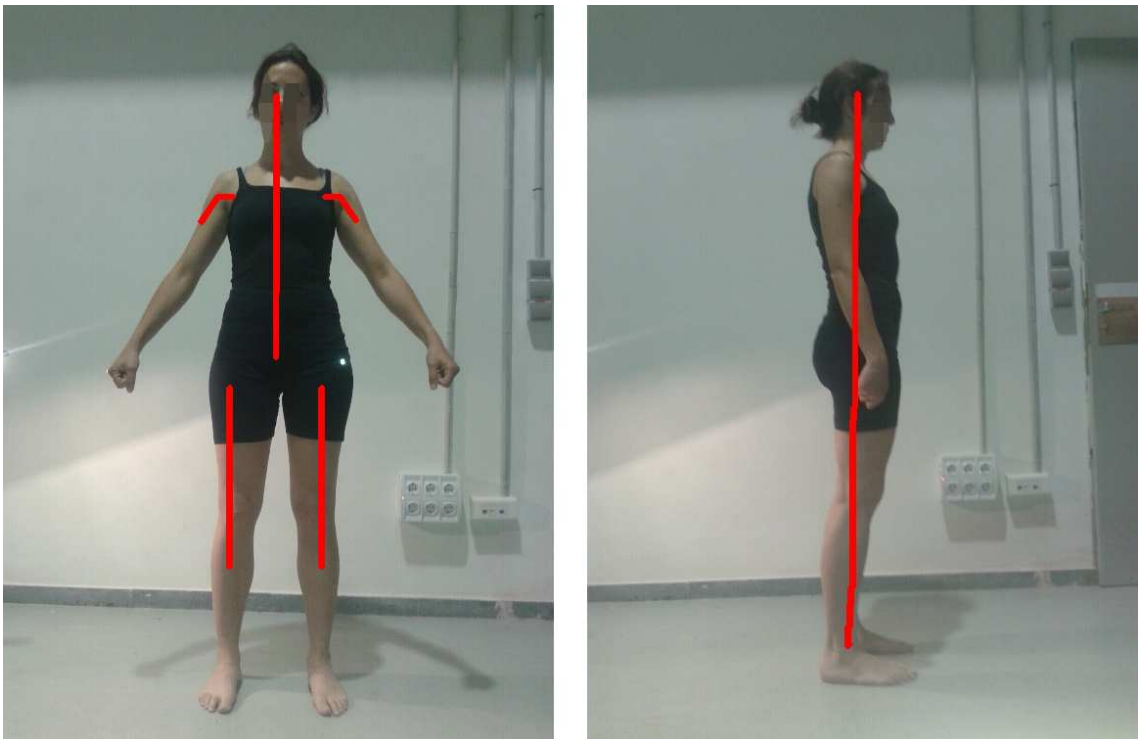


Figura 48. Marcadores para foreground sobre modelo C

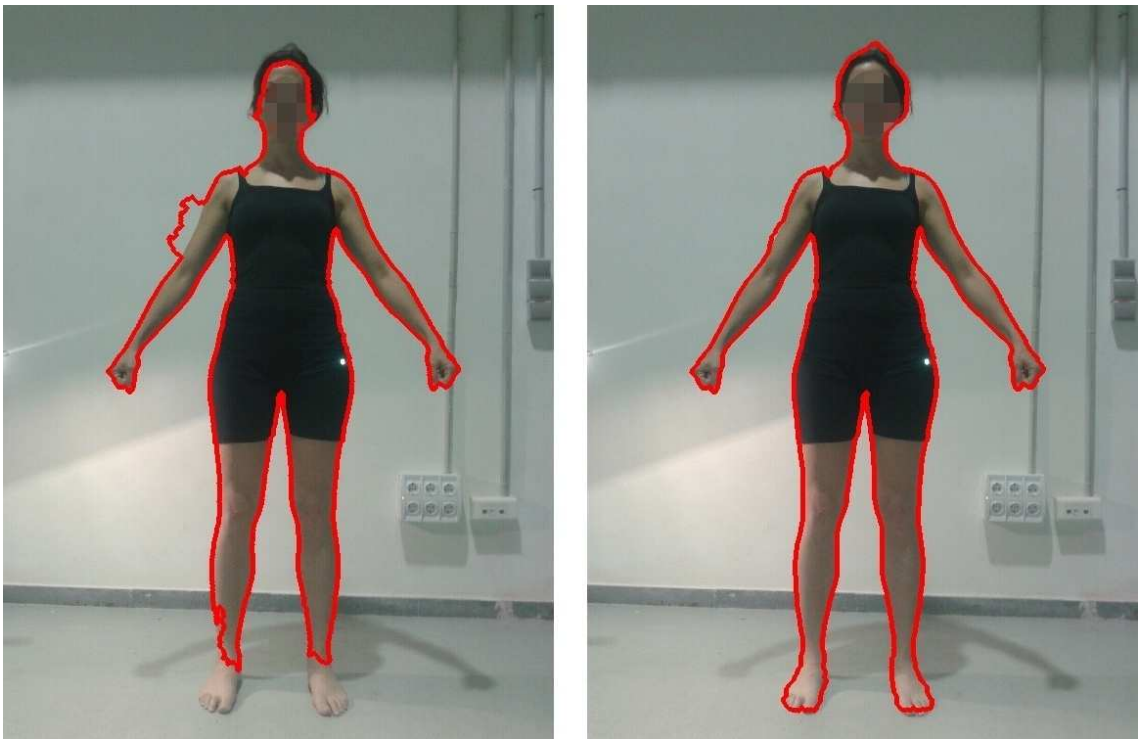


Figura 49. Ejemplo segmentación modelo C posición frontal. Izquierda: watershed. Derecha: grabcut

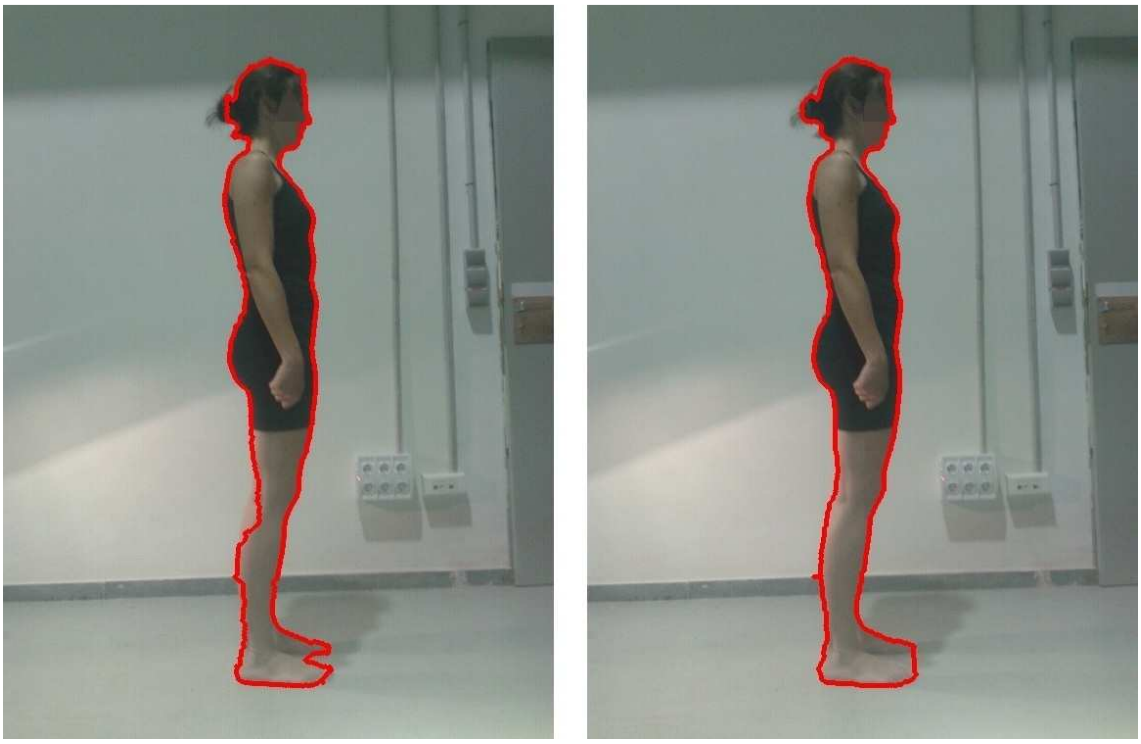


Figura 50. Ejemplo segmentación modelo C posición lateral. Izquierda: watershed. Derecha: grabcut

En las figuras anteriores puede verse que, como en el ejemplo del principio los mejores resultados se obtienen usando el algoritmo de grabcut. En la figura 46 derecha puede observarse

cómo el contorno de la silueta se desvía un poco en una de las piernas. Eso puede deberse a una semejanza del color de la camiseta con el bordillo del suelo, pero como no supone mucha desviación no afectará al software de reconstrucción del modelo 3D. Por lo general si el contorno no se desvía mucho no tendría por qué afectar a dicho software.

Podemos concluir en esta parte del documento que el algoritmo elegido para hacer la segmentación será el algoritmo de grabcut, ya que para obtener ambos contornos el resultado obtenido es mucho más preciso que con el algoritmo de watershed ya que aunque en este último el contorno obtenido en la posición lateral es bastante preciso en el contorno frontal es un completo desastre y se necesitan los dos contornos para poder reconstruir el modelo 3D.

Como desventaja del algoritmo grabcut hay que comentar que es más lento que el algoritmo de watershed y consume más recursos (memoria RAM), pero no dejan de ser unos pocos segundos más que watershed y el resultado es mucho mejor que en este, tal y como se ha demostrado en este capítulo. Por tanto sólo que da concluir que el algoritmo utilizado en la aplicación final será el de grabcut ya que es el que mejor resultados ofrece.

Capítulo 6: Evaluación

Una vez obtenido el fichero con los contornos de frente y de perfil ya se puede reconstruir el modelo 3D (figura 51 y 52). Después ya se pueden empezar a calcular los datos antropométricos que necesitemos. Para saber si los datos antropométricos obtenidos son correctos o si tienen mucho error, se van a comparar con los datos obtenidos con el modelo del cuerpo escaneado con un escáner 3D.

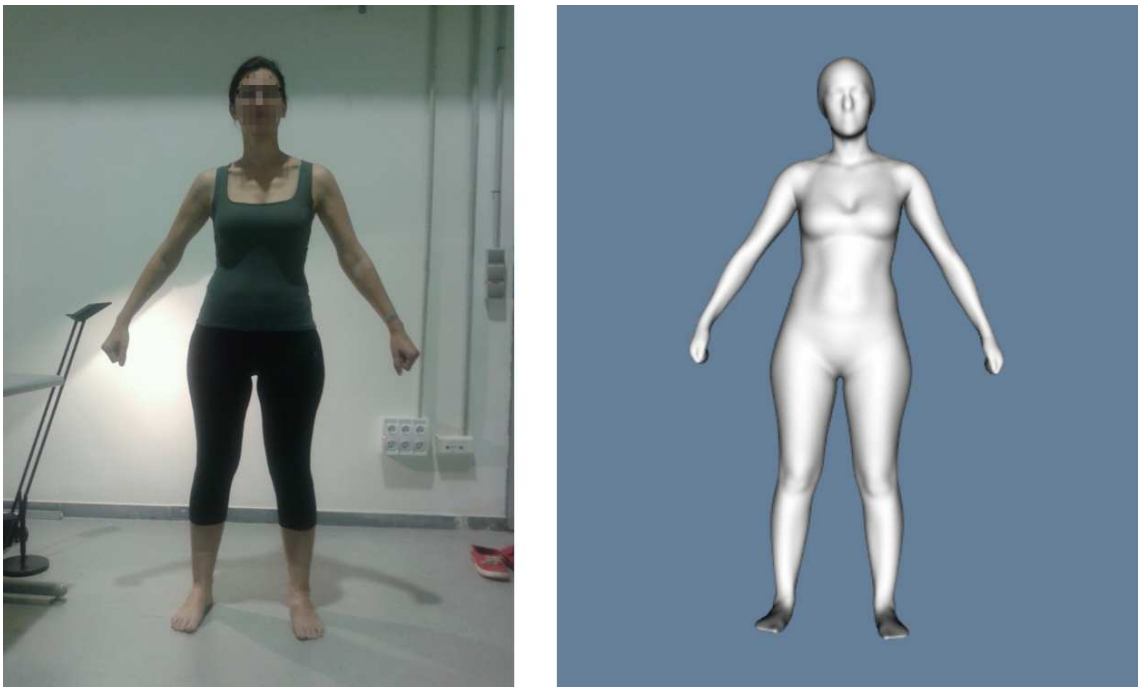


Figura 51. Modelo 3D reconstruido. Vista frontal

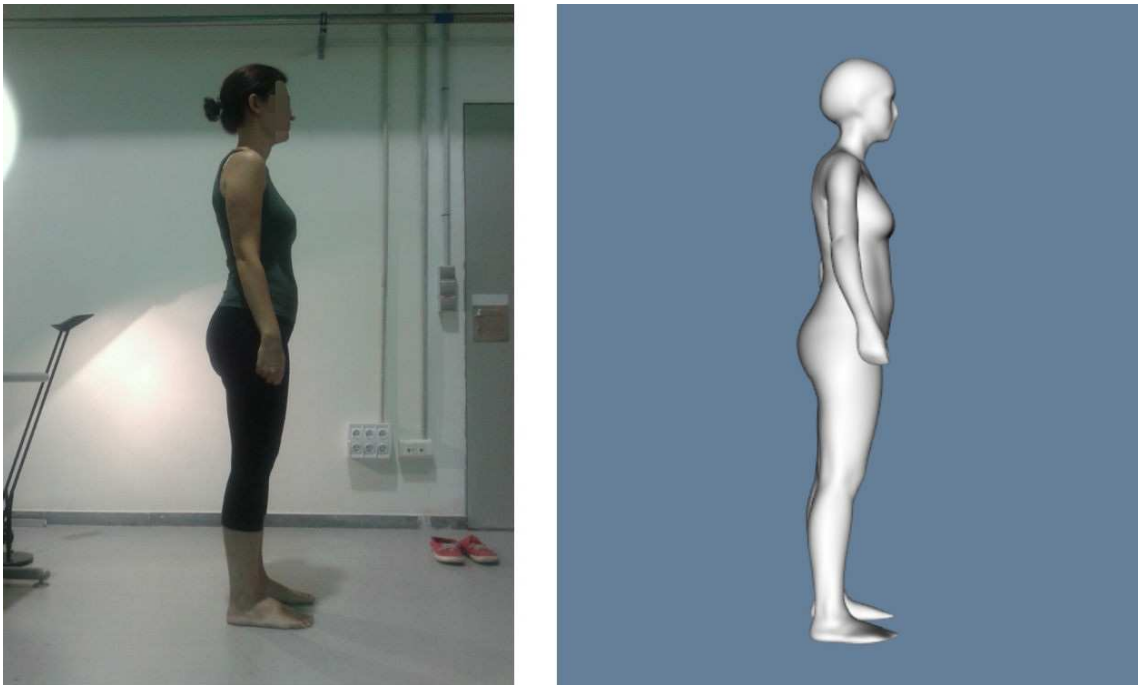


Figura 52. Modelo 3d reconstruido. Vista lateral

Para cada una de las chicas se hizo en los laboratorios del IBV un escáner de cuerpo completo y se calcularon todas las medidas antropométricas posibles mediante un software desarrollado en el IBV.

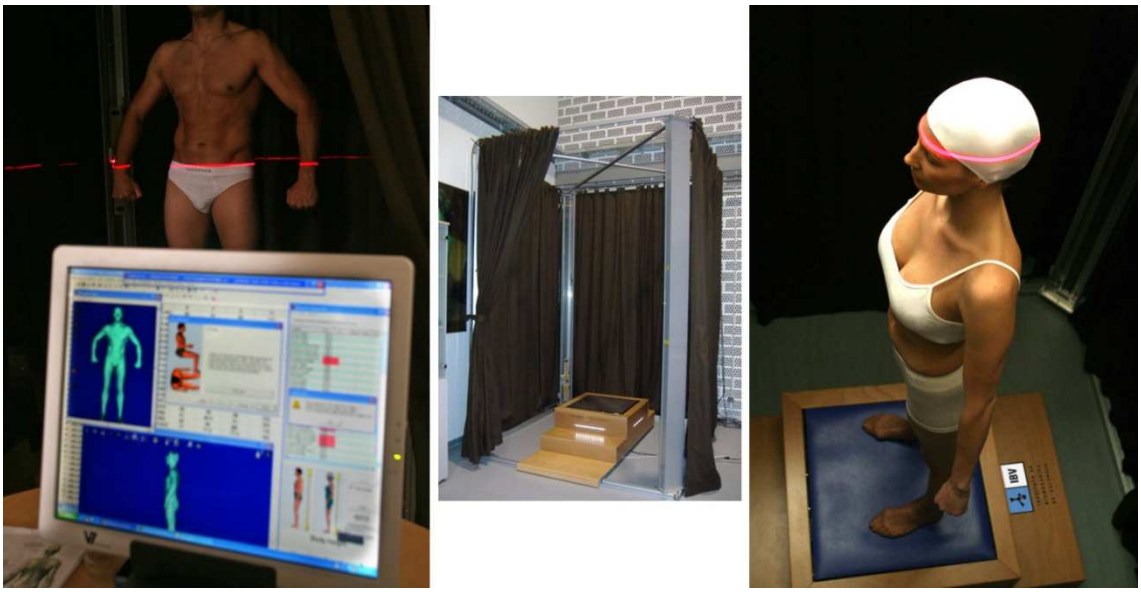


Figura 53. Centro: Escáner 3D de cuerpo completo del IBV. Izquierda y derecha: Ejemplo de escaneo. Fuente: IBV

Entre todas las medidas que se pueden obtener se mostrará sólo algunas de las más representativas para confección de ropa (se pueden calcular más de 200). Estas medidas pueden verse en la siguiente figura para poder ver y entender mejor que zona del cuerpo estamos midiendo.

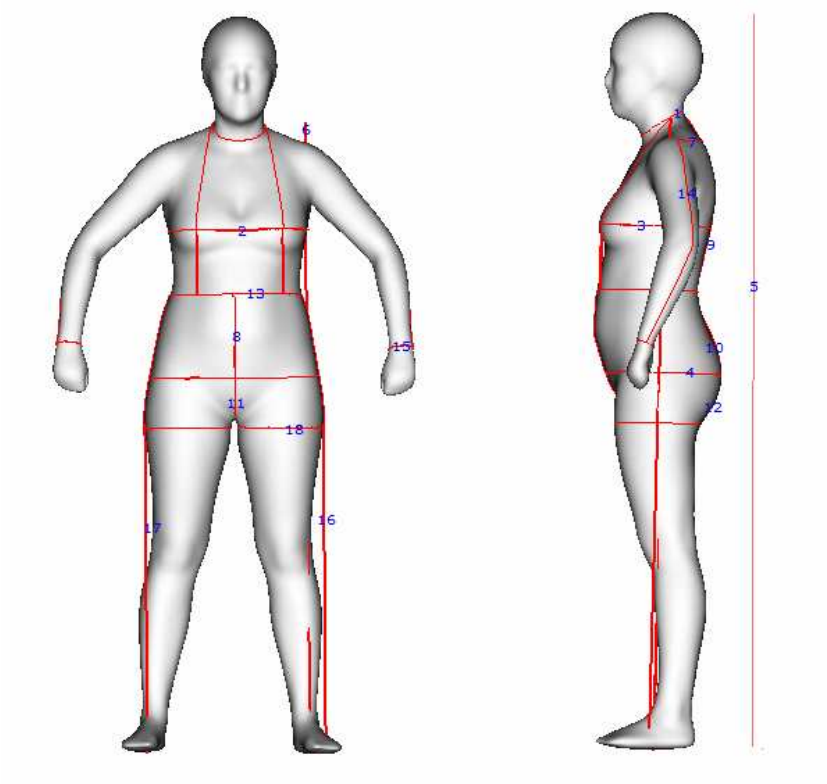


Figura 54. Figuras de frente y perfil con las medidas marcadas. Fuente: IBV

1. Circunferencia a mitad del cuello
2. Anchura del busto
3. Circunferencia del busto / pecho
4. La circunferencia de los glúteos
5. Altura
6. Altura a mitad del cuello
7. Anchura de los hombros horizontal entre acromions
8. Longitud de la cintura frontal
9. Largo de la espalda a la cintura
10. Longitud desde la cintura hasta la nalga
11. Longitud de la entrepierna delantera
12. Longitud de la entrepierna trasera
13. La circunferencia de la cintura
14. Longitud del brazo
15. La circunferencia de la muñeca
16. Longitud de costura lateral hasta el suelo
17. Longitud de costura lateral hasta el tobillo
18. Circunferencia del muslo

Figura 55. Nombre de las medidas numeradas en la figura 54

La definición de todas estas medidas pueden verse en las normas ISO siguientes, ISO 8559:1989: Garment construction and anthropometric surveys -- Body dimensions y ISO 7250-1:2008: Basic human body measurements for technological design -- Part 1: Body measurement definitions and landmarks.

A continuación vamos a mostrar la comparación de los datos antropométricos obtenidos para cada una de las tres chicas, así como la diferencia entre los datos antropométricos obtenidos del modelo escaneado y del modelo reconstruido y el porcentaje de error. Se mostraran en una tabla las medidas calculadas.

Medida	Escaneado (mm)	Modelo reconstruido (mm)	Diferencia (mm)	Error
Circunferencia a mitad del cuello	336,41	324,569	11,841	3,52%
Anchura del busto	193,962	180,723	13,239	6,83%
Circunferencia del busto / pecho	923,11	905,724	17,386	1,88%
La circunferencia de los glúteos	1087,29	1053,22	34,07	3,13%
Altura	1676,7	1657,93	18,77	1,12%
Altura a mitad del cuello	1431,07	1436,18	5,11	0,36%
Anchura de los hombros horizontal entre acromios	383,069	387,796	4,727	1,23%
Longitud de la cintura frontal	463,898	416,788	47,11	10,16%
Largo de la espalda a la cintura	417,75	405,149	12,601	3,02%
Longitud desde la cintura hasta la nalga	200,695	205,607	4,912	2,45%
Longitud de la entrepierna delantera	273,767	351,377	77,61	28,35%
Longitud de la entrepierna trasera	459,643	423,545	36,098	7,85%
La circunferencia de la cintura	801,43	757,944	43,486	5,43%
Longitud del brazo	556,025	569,594	13,569	2,44%
La circunferencia de la muñeca	152,62	154,951	2,331	1,53%
Longitud de costura lateral hasta el suelo	1044,24	1052,57	8,33	0,80%
Longitud de costura lateral hasta el tobillo	971,423	969,711	1,712	0,18%
Circunferencia del muslo	662,885	631,312	31,573	4,76%

Figura 56. Medidas antropométricas modelo A

Medida	Escaneado (mm)	Modelo reconstruido (mm)	Diferencia (mm)	Error
Circunferencia a mitad del cuello	322,498	368,564	46,066	14,28%
Anchura del busto	184,68	190,086	5,406	2,93%
Circunferencia del busto / pecho	951,706	952,906	1,2	0,13%
La circunferencia de los glúteos	1005,65	1053,85	48,2	4,79%
Altura	1698,36	1695,13	3,23	0,19%
Altura a mitad del cuello	1445,99	1443,78	2,21	0,15%
Anchura de los hombros horizontal entre acromios	368,237	391,407	23,17	6,29%
Longitud de la cintura frontal	471,853	455,796	16,057	3,40%
Largo de la espalda a la cintura			0	
Longitud desde la cintura hasta la nalga	179,379	196,631	17,252	9,62%
Longitud de la entrepierna delantera	247,094	349,75	102,656	41,55%
Longitud de la entrepierna trasera	414,237	422,891	8,654	2,09%
La circunferencia de la cintura	809,15	851,921	42,771	5,29%
Longitud del brazo	558,298	547,558	10,74	1,92%
La circunferencia de la muñeca	155,99	162,947	6,957	4,46%
Longitud de costura lateral hasta el suelo	1055,34	1054,8	0,54	0,05%
Longitud de costura lateral hasta el tobillo	978,065	971,627	6,438	0,66%
Circunferencia del muslo	632,39	600,887	31,503	4,98%

Figura 57. Medidas antropométricas modelo B

Medida	Escaneado (mm)	Modelo reconstruido (mm)	Diferencia (mm)	Error
Circunferencia a mitad del cuello	311,559	322,704	11,145	3,58%
Anchura del busto	175,595	160,871	14,724	8,39%
Circunferencia del busto / pecho			0	
La circunferencia de los glúteos	963,637	954,561	9,076	0,94%
Altura	1645,52	1619,23	26,29	1,60%
Altura a mitad del cuello	1402,49	1403,95	1,46	0,10%
Anchura de los hombros horizontal entre acromios	335,645	366,452	30,807	9,18%
Longitud de la cintura frontal	436,679	416,846	19,833	4,54%
Largo de la espalda a la cintura	386,337	404,609	18,272	4,73%
Longitud desde la cintura hasta la nalga	188,47	189,286	0,816	0,43%
Longitud de la entrepierna delantera	232,005	318,824	86,819	37,42%
Longitud de la entrepierna trasera	413,607	387,525	26,082	6,31%
La circunferencia de la cintura	720,615	736,548	15,933	2,21%
Longitud del brazo	562,64	540,406	22,234	3,95%
La circunferencia de la muñeca	155,713	151,509	4,204	2,70%
Longitud de costura lateral hasta el suelo	1042,15	1020,48	21,67	2,08%
Longitud de costura lateral hasta el tobillo	987,079	942,842	44,237	4,48%
Circunferencia del muslo	629,76	546,183	83,577	13,27%

Figura 58. Medidas antropométricas modelo C

Las medidas seleccionas para mostrar en este estudio son las mínimas que se consideran para confeccionar ropa y aunque en las figuras 57 y 58 no se han podido obtener dos de ellas se muestran aquí porque tienen cierta relevancia. Comparando los modelos escaneados de los reconstruidos podemos observar que hay diferencias más pequeñas y otras más grandes, por ejemplo la longitud de la entrepierna delantera, que en los tres casos el error roza el 30-40% y otros casos en los que el error no llega al 1% como por ejemplo la altura a mitad del cuello. Otro dato importante como es la altura se puede observar que el resultado es muy preciso ya que el error es de un 1,12% (18,77 mm) en la figura 56, 0,19% (3,23 mm) en la figura 57 y 1,60 (26,29 mm) en la figura 58. La distancia entre los acromions (figura 59), que suele ser difícil de aproximar, es bastante precisa sobre todo en primer caso, teniendo un error de 1,23 % (4,727 mm). La mayoría al tratarse de diferencias que son de unos pocos milímetros pueden incluso despreciarse y no tenerse en cuenta.

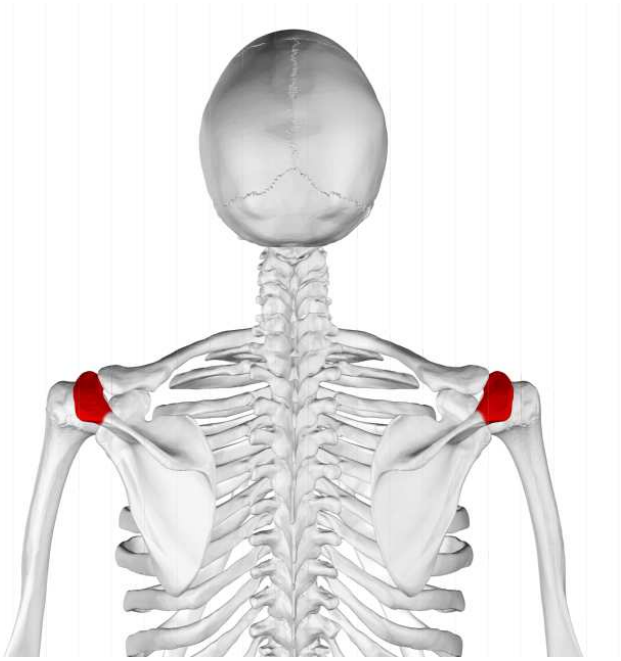


Figura 59. Acromion. Fuente: wikipedia

En cuanto al error medio y la distancia media podemos decir que son errores muy bajos ya que en ninguno de los tres casos se llega al 10% del total y que son errores asumibles que también se introducen tomando medidas de otra forma, ya sea tomando los datos de forma manual o a partir de imágenes 2D como hemos visto en el apartado de antropometría.

	Diferencia media (mm)	Error medio
Modelo A	21,95324815	4,96%
Modelo B	20,725	6,05%
Modelo C	24,28772222	6,23%

Figura 60. Media de la diferencia y el error medio

Hay que tener en cuenta que el software de cálculo de medidas antropométricas está aún en desarrollo y está en su primera versión (hay algunas medidas que no se han salido bien y no se han puesto), pero como se ha dicho antes la toma de medidas antropométricas suele introducir errores que son asumibles, por lo que podemos decir que los cálculos son muy buenos y además aún hay mucho margen de mejora.

Capítulo 7: Conclusiones y Proyectos futuros.

En el presente proyecto se ha realizado una aplicación en Android que sirve de nexo entre el usuario que realiza las fotos y un futuro servidor que se encargará de realizar la reconstrucción 3D y de obtener las medidas antropométricas del cuerpo del usuario (mediante algoritmos desarrollados en el IBV) a partir de los contornos obtenidos de las fotografías tomadas de frente y de perfil del usuario.

Se han presentado todos los materiales empleados haciendo especial hincapié en la librería OpenCV que se encarga de la parte de tratamiento de imagen y la plataforma Android donde se desarrolla toda la aplicación.

Los objetivos planteados al principio del proyecto se han cumplido ya que se ha demostrado que un teléfono móvil, un smartphone en este caso, tiene potencia suficiente para realizar cálculos complejos como puede ser la segmentación de una imagen a partir de unos marcadores y como estos contornos calculados son validos para que se pueda reconstruir (usando los algoritmos desarrollados en el IBV) un modelo 3D del cuerpo y obtener medidas antropométricas relevantes en el mundo de la moda para venta online, que como hemos visto sufre grandes pérdidas debidas a las devoluciones por tallas erróneas.

Se ha ido viendo a lo largo del documento que poder mostrar por la pantalla del dispositivo los resultados de los cálculos realizados servirá para ahorrar tiempo para recibir los datos del servidor ya que se reduce la carga de trabajo en este y no hay que entrar en un bucle enviar-recibir hasta que los resultados del usuario sean correctos.

Se ha visto que los datos almacenados ocupan muy poco (sobre 15KB), ya que sólo se tienen que almacenar los datos relacionados con los contornos calculados y no las imágenes del usuario, garantizando así su privacidad y consiguiéndose también un ahorro en ancho de banda, importante si usamos tarifa de datos móvil, y tiempo de subida.

Hemos podido comprobar cómo las medidas antropométricas calculadas del modelo reconstruido a partir de los perfiles de las dos fotografías se asemejan bastante a la realidad pudiendo obtenerse muchas medidas de forma rápida (más de 200 medidas).

Dados los buenos resultados obtenidos se puede concluir que la base para futuras líneas de trabajo es positiva, ya que se ha demostrado que a partir de dos fotografías tomadas con un dispositivo que actualmente tiene casi todo el mundo, como es un smartphone, se puede tener un sistema sencillo de manejar para el cliente, con el que puede tener su propio modelo 3D con todas las medidas antropométricas necesarias para la compra online de ropa sin necesidad de tener que tomar medidas en su propia casa ni tener que depender de experiencias previas con las marcas de ropa, ya que como hemos visto cada una tiene su propio sistema de tallas e incluso dentro de cada marca puede cambiar de una línea de productos a otra.

Como proyectos futuros hay que destacar que el algoritmo de cálculo de datos antropométricos puede mejorar, ya que la versión disponible a la hora de hacer este proyecto era la primera. También se pueden implementar mejores algoritmos de tratamiento de imagen para que se pueda hacer una mejor segmentación en entornos más complejos donde sea más dificil extraer siluetas, ya que, por experiencia, el cliente no va saber nada de tratamiento de imagen ni le va a

importar, sólo querrá resultados rápidos y buenos sin que tenga que trabajar, ni pensar ni leer mucho.

Cabe destacar que en la aplicación se tiene que desarrollar el sistema de envio de datos al servidor, ya que no se ha tratado al no estar el servidor aún montado, pero es un tema en el que Android también tiene varias alternativas, como puede ser el uso de sockets o el protocolo HTTP.

En la parte de venta online se pueden desarrollar armarios virtuales con el modelo reconstruido y aplicar técnicas de ajuste de ropa al modelo 3D que simulen la interacción de los diferentes tejidos sobre el cuerpo humano. También se pueden aplicar técnicas de minería de datos para desarrollar patrones de búsqueda de los gustos referidos a los clientes.

La aplicación puede desarrollarse en otros sistemas como iOS o Windows Phone, aunque habría que rehacerla por completo, ya que los lenguajes de programación son diferentes a Android (principalmente Objective-C en iOS y C# entre otros en Windows Phone). También puede desarrollarse para otros dispositivos como tabletas si estas incluyen cámara trasera. En Android para pasar la aplicación de un teléfono a una tableta se necesitan muy pocos cambios, bastaría con ajustar la interfaz del usuario para que funcione en otros tamaños de pantalla y resolución.

Capítulo 8: Bibliografía

[1] http://www.emarketer.com/Article/Global-B2C-Ecommerce-Sales-Hit-15-Trillion-This-Year-Driven-by-Growth-Emerging-Markets/1010575

[2] http://www.ystats.com/en/reports/preview.php?reportId=1038&start=0

[3] http://www.europapress.es/economia/noticia-economia-moda-impulsa-crecimiento-commerce-50-espanoles-reconoce-comprar-ropa-internet-20130828112114.html

[4] http://www.marketingdirecto.com/actualidad/digital/la-moda-lleva-en-volandas-el-crecimiento-del-e-commerce-a-nivel-global/

[5] http://blogs.wsj.com/tech-europe/2013/05/29/online-fashion-rewards-set-to-continue-says-report/?mod=WSJBlog&mod=

[6] http://www.insht.es/Ergonomia2/Contenidos/Promocionales/Diseno%20del%20puesto/DTEAntropometriaDP.pdf

[7] Pheasant S. Bodyspace. Antropometry, Ergonomics and Design. London: Taylor & Francis. ISBN:0850663520.

[8] http://es.wikipedia.org/wiki/Antropometr%C3%ADa

[9] Veitch, D.; Caple, D.; Blewett, V. Sizing Up Australia: How contemporary is the anthropometric data Australian designers use?. Australian Safety and Compensation Council. January 2009.

[10] http://www.ugr.es/~jhuertas/EvaluacionFisiologica/Antropometria/antropintro.htm

[11] http://ocw.upm.es/educacion-fisica-y-deportiva/kinantropometria/contenidos/temas/Tema-2.pdf

[12] Simmons, K. P. Body measurement techniques: A comparison of three-dimensional body scanning and physical anthropometric methods. Ph.D. degree in Textile Technology and Management, 2001.

[13] http://www.symcad.com/eng/index.htm

[14] http://www.tc2.com

[15] http://www.cyberware.com/

[16] http://www.human-solutions.com

[17] http://es.wikipedia.org/wiki/Esc%C3%A1ner_3D

[18] Hin, A. J. S. and A. Krul (2005). Performance of Human Solutions body dimension analysis software. Sosterberg, The Netherlands Organization (TNO).

[19] Robinette, K.; Blackwell, S.; Daanen, H.; Fleming, Boehmer, M.; Brill, T.; Hoeferlin, D. and Burnsides, D. (2002). Civilian American and European Surface Anthropometry Resource

(CAESAR), Final Report, Volume I: Summary. AFRL-HE-WP-TR-2002-0169, United States Air Force Research Laboratory, Human Effectiveness Directorate, Crew System Interface Division. 2255 H Street, Wright-Patterson AFB OH 45433-7022.

[20] Bougourd, J. and Treleaven, P. UK National Sizing Survey – SizeUK. International Conference on 3D Body Scanning Technologies, Lugano, Switzerland, 19-20 October 2010.

[21] http://www.whatfitsme.com

[22] http://www.sizewand.com

[23] http://mipso.me

[24] http://www.truefit.com/login

[25] http://www.zafu.com

[26] www.me-ality.com/

[27] http://www.upcload.com/

[28] http://www.poikos.com/our-platform/flixfit/

[29] http://www.fits.me/

[30] http://www.styku.com/business/

[31] http://www.bodypal.com/

[32] Robinette, K. M. and J. T. McConville (1982). An Alternative to Percentile Models: SAE Technical Paper 810217. 1981 SAE Transactions. Warrendale, PA, Society of Automotive Engineers: 938-946.

[33] Diffrient, N., Tilley, A. R., & Bardagjy, J. C. (1983). Humanscale. New York: MIT Press.

[34] http://www.human-solutions.com/mobility/front_content.php?idcat=250&lang=8

[35] Molenbroek, J.F.M. and Bruin, R. de, 2005. Enhancing the use of anthropometric data. In: Human factors in design, safety, and management. Delft, 2005. Maastricht: Shaker Publishing.

[36] http://www.bbc.co.uk/mundo/noticias/2014/01/140121_tecnologia_maniquis_virtuales_il.shtml

[37] http://corpo.myvirtualmodel.com/index.html

[38] http://www.digitalfashion.jp/new/product/haoreba/

[39] http://www.sunfeet.es/es/

[40] http://www.masquenegocio.com/2014/02/05/radiografia-mercado-movil-espana-iab/

[41] http://www.iabspain.net/wp-content/uploads/downloads/2013/09/V_Estudio_Mobile_Marketing_version_corta.pdf

[42] http://developer.android.com/

[43] El gran libro de Android. Jesús Tomás Gironés.3ª Edición

[44] http://es.wikipedia.org/wiki/Android

[45] http://beta.appinventor.mit.edu/

[46] http://opencv.org/

[47] Meyer, F. and Beucher, S. [1990]. “Morphological Segmentation”, J. Visual Comm., and Image Representation vol. 1, no. 1, pp. 21-46.

[48] http://research.microsoft.com/pubs/67890/siggraph04-grabcut.pdf

[49] Amstutz, E., Teshima, T., Kimura, M., Mochimaru, M. and Saito, H. [July 2008] “PCA-based 3D Shape Reconstruction of Human Foot Using Multiple Viewpoint Cameras”, International Journal of Automation and Computing.

[50] http://docs.opencv.org/

[51] http://www.samsung.com/es/support/model/GT-I8160OKAXEC-techspecs

Capítulo 9: Anexos

En los apartados siguientes se adjunta el código de las diversas partes que forman la aplicación.

9.1. Anexo A: Clases

SplashScreenActivity.java

Esta clase se encarga de, como se nombre indica, mostrar un splash screen por pantalla durante unos pocos segundos el título o logo de la aplicación.

```
package org.example.proyectofinal;

import java.util.Timer;
import java.util.TimerTask;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;

public class SplashScreenActivity extends Activity {

  private long splashDelay = 4000; //4 segundos

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.splash_screen);

    TimerTask task = new TimerTask() {
      @Override
      public void run() {
        Intent mainIntent = new Intent().setClass(SplashScreenActivity.this,
MainActivity.class);
        startActivity(mainIntent);
        finish();//Destruimos esta activity para prevenir que el usuario retorne aquí
presionando el boton Atras.
      }
    };

    Timer timer = new Timer();
    timer.schedule(task, splashDelay);//Pasado los 4 segundos dispara la tarea
  }

}
```

MainActivity.java

Esta clase se encarga de mostrar el tutorial al usuario y de manejar todos las vistas que se utilizan para montar el tutorial y de lanzar la clase que se encarga del manejo de la cámara.

```
package org.example.proyectofinal;



import org.opencv.android.BaseLoaderCallback;
import org.opencv.android.LoaderCallbackInterface;
import org.opencv.android.OpenCVLoader;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.view.animation.AnimationUtils;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ViewFlipper;
```

```
public class MainActivity extends Activity {

        ViewFlipper vf;
        float init_x;
        ProgressBar pb;
        TextView txt;
        EditText aTxt;
        EditText eTxt;
        EditText pTxt;
        private static final String TAG = "MainActivity";
        int cont3=0, cont4=0, cont5=0;
        TextView txtPaso3, txtPaso4, txtPaso5;

        private BaseLoaderCallback  mLoaderCallback = new BaseLoaderCallback(this) {
         @Override
         public void onManagerConnected(int status) {
             switch (status) {
                 case LoaderCallbackInterface.SUCCESS:
                 {
                     Log.i(TAG, "OpenCV loaded successfully");

                 } break;
                 default:
                 {
                     super.onManagerConnected(status);
                 } break;
             }
         }
    };

    static {
         if (!OpenCVLoader.initDebug()) {
             // Handle initialization error
         }
    }

        @Override
        protected void onCreate(Bundle savedInstanceState) {

         Log.i(TAG, "Trying to load OpenCV library");
         if (!OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_2_4_6, this,
mLoaderCallback))
         {
           Log.e(TAG, "Cannot connect to OpenCV Manager");
         }
         else{ Log.i(TAG, "opencv successfully added"); }

                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);

                txtPaso3=(TextView)findViewById(R.id.txtPasotres);
                txtPaso3.setText(R.string.pasotres);
                txtPaso3.setCompoundDrawablesWithIntrinsicBounds(0,
R.drawable.frente_con_marcador_1_c, 0, 0);

                txtPaso4=(TextView)findViewById(R.id.txtPasocuatro);
                txtPaso4.setText(R.string.pasocuatro);

                txtPaso5=(TextView)findViewById(R.id.txtPasocinco);
                txtPaso5.setText(R.string.pasocinco);
                txtPaso5.setCompoundDrawablesWithIntrinsicBounds(0,
R.drawable.perfil_original_1, 0, 0);

                vf=(ViewFlipper)findViewById(R.id.viewFlipper);

                vf.setOnTouchListener(new ListenerTouchViewFlipper());

                pb=(ProgressBar)findViewById(R.id.progressBar1);
                txt=(TextView)findViewById(R.id.txtView);


                pb.setProgress(0);
                txt.setText("1/7");
        }


        private class ListenerTouchViewFlipper implements View.OnTouchListener{
```

```
        @Override
        public boolean onTouch(View v, MotionEvent event) {

            switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                init_x=event.getX();
                return true;
            case MotionEvent.ACTION_UP:
                float distance =init_x-event.getX();




                if(distance>0)
                {
                      int displayedChild = vf.getDisplayedChild();

                    if (displayedChild==6)

                      break;

                    if ((displayedChild == 3) && (cont3==0)){

                      txtPaso3.setText(R.string.pasotres_uno);
                      txtPaso3.setCompoundDrawablesWithIntrinsicBounds(0,
R.drawable.frente_original_1, 0, 0);
                      cont3=1;
                      break;

                    }

                    if ((displayedChild == 3) && (cont3==1)){

                      txtPaso3.setText(R.string.pasotres_dos);
                      txtPaso3.setCompoundDrawablesWithIntrinsicBounds(0,
R.drawable.originalwfrente, 0, 0);
                      cont3=2;
                      break;

                      }

                    if ((displayedChild == 4) && (cont4==0)){

                              txtPaso4.setText(R.string.pasocuatro_uno);
                              cont4=1;
                              break;

                    }


                    if ((displayedChild == 5) && (cont5==0)){

                              txtPaso5.setText(R.string.pasocinco_uno);
                              txtPaso5.setCompoundDrawablesWithIntrinsicBounds(0,
R.drawable.originalwperfil, 0, 0);
                              cont5=1;
                              break;

                    }

                    if ((displayedChild == 5) && (cont5==1)){

                              txtPaso5.setText(R.string.pasocinco_dos);
                              txtPaso5.setCompoundDrawablesWithIntrinsicBounds(0,
R.drawable.originalwperfil, 0, 0);
                              cont5=2;
                              break;

                    }


                     vf.setInAnimation(AnimationUtils.loadAnimation(v.getContext(),
R.anim.slide_left));
```

```
                        vf.showNext();

                        int child = vf.getDisplayedChild();
                        pbchange(child);

                }

                if(distance<0)
                {

                      int displayedChild = vf.getDisplayedChild();

                    if (displayedChild == 0)

                            break;

                    if ((displayedChild == 3) && (cont3==2)){

                      txtPaso3.setText(R.string.pasotres_uno);
                      txtPaso3.setCompoundDrawablesWithIntrinsicBounds(0,
R.drawable.frente_original_1, 0, 0);
                            cont3=1;
                            break;

                    }

                    if ((displayedChild == 3) && (cont3==1)){

                      txtPaso3.setText(R.string.pasotres);
                      txtPaso3.setCompoundDrawablesWithIntrinsicBounds(0,
R.drawable.frente_con_marcador_1_c, 0, 0);
                            cont3=0;
                            break;

                    }


                    if ((displayedChild == 4) && (cont4==1)){

                            txtPaso4.setText(R.string.pasocuatro);
                            txtPaso5.setCompoundDrawablesWithIntrinsicBounds(0,
R.drawable.perfil_con_marcador_1, 0, 0);
                            cont4=0;
                            break;

                    }


                    if ((displayedChild == 5) && (cont5==2)){

                            txtPaso5.setText(R.string.pasocinco_uno);
                            txtPaso5.setCompoundDrawablesWithIntrinsicBounds(0,
R.drawable.originalwperfil, 0, 0);
                            cont5=1;
                            break;

                    }

                    if ((displayedChild == 5) && (cont5==1)){

                            txtPaso5.setText(R.string.pasocinco);
                            txtPaso5.setCompoundDrawablesWithIntrinsicBounds(0,
R.drawable.perfil_original_1, 0, 0);
                            cont5=0;
                            break;

                    }



                     vf.setInAnimation(AnimationUtils.loadAnimation(v.getContext(),
R.anim.slide_right));

                        vf.showPrevious();

                        int child = vf.getDisplayedChild();
                        pbchange(child);
```

```
                }

            default:
                break;
            }
            return false;
        }
    }


        public void pbchange(int child){

                        if(child==0) {
                          pb.setProgress(0);
                           txt.setText("1/7");
                        }

                        if(child==1) {
                          pb.setProgress(1);
                          txt.setText("2/7");
                        }

                       if(child==2){
                         pb.setProgress(2);
                         txt.setText("3/7");
                       }

                    if(child==3) {
                           pb.setProgress(3);
                           txt.setText("4/7");
                    }

                    if(child==4) {
                           pb.setProgress(4);
                           txt.setText("5/7");
                    }

                    if(child==5) {
                           pb.setProgress(5);
                           txt.setText("6/7");
                    }

                    if(child==6) {
                           pb.setProgress(6);
                           txt.setText("7/7");
                    }

        }

    public void lanzarEmpezar(View view){
                aTxt=(EditText)findViewById(R.id.TxtAltura);
                eTxt=(EditText)findViewById(R.id.TxtEdad);
                pTxt=(EditText)findViewById(R.id.TxtPeso);
                String a=aTxt.getText().toString();
                String e=eTxt.getText().toString();
                String p=pTxt.getText().toString();
                double altura=0,peso=0;
                int edad=0;

                if(p.matches("")==false){
                        peso=Double.parseDouble(p);
                }

                if(e.matches("")==false){
                        edad=Integer.parseInt(e);
                }

        if(a.matches("")==true){
                Toast.makeText(this, "El campo altura esta vacio",
Toast.LENGTH_SHORT).show();
                return;
        }

        if(a.matches("")==false){
                altura=Double.parseDouble(a);
```

```
        }

        if (altura>2.5 | altura<=1){
                Toast.makeText(this, "El campo altura tiene que estar entre 1 y 2.5",
Toast.LENGTH_SHORT).show();
                return;
        }
                Intent i = new Intent(this, CameraMain.class);
                i.putExtra("altura", altura);
                i.putExtra("peso", peso);
                i.putExtra("edad", edad);
                startActivityForResult(i,1);

      }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode==1 && resultCode==RESULT_OK) {
                finish();
         }
         super.onActivityResult(requestCode, resultCode, data);
    }

}
```

CameraMain.java

Esta clase se encarga de poner en funcionamiento la cámara, de hacer la segmentación, de mostrar por pantalla los resultados de esta, de controlar los sensores, de mostrar por pantalla un mensaje que indica que se está calculando la segmentación, de almacenar las fotos necesarias para ver el proceso de la segmentación y de montar el fichero xml con los datos necesarios sobre la cámara y de las matrices de rotación y de los contornos de ambas fotos.

```
package org.example.proyectofinal;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;


import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.opencv.android.Utils;
import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.MatOfPoint;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.highgui.Highgui;
import org.opencv.imgproc.Imgproc;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.hardware.Camera;
import android.hardware.Camera.AutoFocusCallback;
import android.hardware.Camera.PictureCallback;
import android.hardware.Camera.ShutterCallback;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
```

```
import android.annotation.TargetApi;
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnTouchListener;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.FrameLayout;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

@TargetApi(Build.VERSION_CODES.GINGERBREAD)
public class CameraMain extends Activity implements OnTouchListener,SensorEventListener{

        Camera mCamera;
        Preview mPreview;
        int tipoimagen;
        ImageView imgView;
    private static final String TAG ="Mas Pruebas";
    String mPictureFileName;
    byte[] d;
        Button btnYes;
        Button btnNo;
        TextView txtV;
        boolean takepicture;
        private float focalLength=0;
        private float verticalAngle=0;
        private float horizontalAngle=0;
        private SensorManager mSensorManager;
        private Sensor mGravity;
        double gx=0,gy=0,gz=0,g=0,fixsen=0,fizsen=0,Gx=0,Gy=0,Gz=0,fix=0,fiz=0;
        double e01_frente=0,e10_frente=0,e12_frente=0,e21_frente=0;
        double e01_perfil=0,e10_perfil=0,e12_perfil=0,e21_perfil=0;
        String fileName;
        String texto;
        String photoName;

    private Document doc;
    private Element rootElement;

    String fix_frente=new String();
    String fix_perfil=new String();
    String fiz_frente=new String();
    String fiz_perfil=new String();


    String df_frente=new String();
    String ah_frente=new String();
    String av_frente=new String();

    String gx_frente=new String();
    String gy_frente=new String();
    String gz_frente=new String();

    String gx_perfil=new String();
    String gy_perfil=new String();
    String gz_perfil=new String();


        TextView TxtFxF;
        TextView TxtFzF;
        TextView TxtDF;
        TextView TxtAh;
        TextView TxtAv;

        TextView TxtFxP;
        TextView TxtFzP;

        TextView TxtE01f;
```

```
        TextView TxtE10f;
        TextView TxtE12f;
        TextView TxtE21f;

        TextView TxtE01p;
        TextView TxtE10p;
        TextView TxtE12p;
        TextView TxtE21p;

        List<MatOfPoint> contours;
        Mat matContour;
        Mat mRgba=new Mat();

        Button salir;

        ProgressDialog dialog;   //widget que se encarga de mostrar por pantalla en
mensaje de espera

        double valores[]=new double[12];
        String tag=new String();
        String names[]=null;
        String
nfrente[]={"Distancia_Focal","Angulo_Horizontal","Angulo_Vertical","Giro_eje_x_Frente","
Giro_eje_z_Frente","gx_Frente","gy_Frente","gz_Frente"};
        String nperfil[]={"Giro_eje_x_Perfil","Giro_eje_z_Perfil",
"gx_Perfil","gy_Perfil","gz_Perfil"};
        int alt=0;
        String
patch=Environment.getExternalStorageDirectory()+"/Android"+"/data"+"/org.example.proyect
ofinal"+"/files"+"/resultados.xml";
        String dir=new String();
        File myDir;
        boolean existe=false;


        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.camera);
                mCamera=getCameraInstance();
                mPreview = new Preview(this,mCamera);
                mPreview.setOnTouchListener(CameraMain.this);
                FrameLayout preview = (FrameLayout) findViewById(R.id.preview);


                preview.addView(mPreview);

         imgView=(ImageView)findViewById(R.id.imgView);
         imgView.setImageResource(R.drawable.sil1_t);
         btnYes=(Button)findViewById(R.id.btnSi);
         btnNo=(Button)findViewById(R.id.btnNon);
         txtV=(TextView)findViewById(R.id.txtView3);

         dialog = new ProgressDialog(this);
         dialog.setMessage("Espere por favor");
         dialog.setTitle("Procesando");

         btnYes.setEnabled(false);
         btnNo.setEnabled(false);

         tipoimagen=0;

                mPreview.Imagen(tipoimagen);

                mSensorManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);
                mGravity = mSensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY);

                mSensorManager.registerListener(this, mGravity,
SensorManager.SENSOR_DELAY_NORMAL);

                File f = new File(patch);
                if(f.exists()){
                        f.delete();
                        Log.d(TAG, "Fichero borrado");

                }
```

```
        }


    private Camera getCameraInstance() {
        Camera camera = null;

        try {
            camera = Camera.open();
        } catch (Exception e) {
            // cannot get camera or does not exist
        }
        return camera;
    }

         public boolean onTouch(View v, MotionEvent event) {
                 //Se encarga de controlar las pulsaciones sobre la pantalla para
capturar una imagen
                Log.i(TAG,"onTouch event");


                if(existe==false){

       dir=Environment.getExternalStorageDirectory().getAbsolutePath()+"/Android"+"/data
"+"/org.example.proyectofinal"+"/files";

                        myDir=new File(dir);



                        if(!myDir.exists()){
                                myDir.mkdirs();
                        }

                }
                existe=true;

                if(tipoimagen==0){
                       fileName=myDir+"/frente.jpg";

                }else if(tipoimagen==1){
                       fileName=myDir+"/perfil.jpg";
                }

                CameraStartAutoFocus();
                takePicture(fileName);
                mPictureFileName=fileName;
                return false;
            }

            @Override
            public boolean onKeyDown(int keyCode, KeyEvent event) {
                if ((keyCode == KeyEvent.KEYCODE_BACK)) {
                        Intent i = new Intent();
                        setResult(RESULT_OK,i);
                        finish();

                }
                return super.onKeyDown(keyCode, event);
            }



               public void takePicture(final String fileName) {
                Log.i(TAG, "Taking picture");
                this.mPictureFileName = fileName;
                takepicture = true;

                mCamera.autoFocus(myAutoFocusCallback);

               }

               AutoFocusCallback myAutoFocusCallback = new AutoFocusCallback()
               {
                       //Se encarga de lanzar el autofoco de la cámara. Antes hay que
activarlo en la clase Preview.java
                       public void onAutoFocus(boolean arg0, Camera Camera)
```

```
                        {
                                if ( takepicture ){
                                Log.d(TAG, "onAutofocus.");

                        if(tipoimagen==0){
                                fix=(double)Math.round(fixsen*1000)/1000;
                                fiz=(double)Math.round(fizsen*1000)/1000;
                                 fix_frente="Giro eje x: "+fix+" grados";
                                 fiz_frente="Giro eje z: "+fiz+" grados";


                                e01_frente=-(double)Math.round((gx/g)*1000)/1000;
                                e10_frente=(double)Math.round((gx/g)*1000)/1000;
                                e12_frente=-(double)Math.round((gz/g)*1000)/1000;
                                e21_frente=(double)Math.round((gz/g)*1000)/1000;

                                Gx=(double)Math.round(gx*1000)/1000;
                                Gy=(double)Math.round(gy*1000)/1000;
                                Gz=(double)Math.round(gz*1000)/1000;
                                gx_frente="gx: "+Gx;
                                gy_frente="gy: "+Gy;
                                gz_frente="gz: "+Gz;

                        }else if(tipoimagen==1){

                                fix=(double)Math.round((fixsen*1000)/1000);
                                fiz=(double)Math.round((fizsen*1000)/1000);
                                 fix_perfil="Giro eje x: "+fix+" grados";
                                 fiz_perfil="Giro eje z: "+fiz+" grados";


                                 e01_perfil=-(double)Math.round((gx/g)*1000)/1000;
                                    e10_perfil=(double)Math.round((gx/g)*1000)/1000;
                                    e12_perfil=-(double)Math.round((gz/g)*1000)/1000;
                                    e21_perfil=(double)Math.round((gz/g)*1000)/1000;


                                    Gx=(double)Math.round(gx*1000)/1000;
                                    Gy=(double)Math.round(gy*1000)/1000;
                                    Gz=(double)Math.round(gz*1000)/1000;
                                    gx_perfil="gx: "+Gx;
                                    gy_perfil="gy: "+Gy;
                                    gz_perfil="gz: "+Gz;

                        }

                        Camera.takePicture(shutterCallback, rawPictureCallback,
jpegPictureCallback);
                        takepicture = false;

                                }

                                }

                };

                ShutterCallback shutterCallback = new ShutterCallback()
                {
                        public void onShutter()
                        {
                                // Just do nothing.
                        }
                };

                PictureCallback rawPictureCallback = new PictureCallback()
                {
                        public void onPictureTaken(byte[] arg0, Camera arg1)
                        {
                                // Just do nothing.
                        }
                };


                //@Override
                PictureCallback jpegPictureCallback = new PictureCallback()
                {
                public void onPictureTaken(byte[] data, Camera camera) {
```

```
                        // TODO Auto-generated method stub
                    Log.i(TAG, "Saving a bitmap to file");
                    // The camera preview was automatically stopped. Start it again.

                    d=data;
                    Log.d(TAG, "onPictureTaken.");
                    new Task().execute(data);



                   }
                   };


          public void showMessage(String s){
                  //Muestra un mensaje por pantalla cuando se graba la imagen principal
                  Toast.makeText(this, s , Toast.LENGTH_SHORT).show();
          }

                public void CameraStartAutoFocus()
                {
                        Log.d(TAG, "CameraStartAutoFocus.");
                        takepicture = false;
                        mCamera.autoFocus(myAutoFocusCallback);
                }

        public void Disconnect(){
                //Para hacer la animacion de salida y hacer invisibles los elementos que
indican si la foto es correcta o no.
         Animation fadeOutAnimation = AnimationUtils.loadAnimation(CameraMain.this,
R.anim.fade_out);
         btnNo.setVisibility(View.INVISIBLE);
         btnNo.setEnabled(false);
         btnNo.startAnimation(fadeOutAnimation);
         btnYes.setVisibility(View.INVISIBLE);
         btnYes.setEnabled(false);
         btnYes.startAnimation(fadeOutAnimation);

         txtV.setVisibility(View.INVISIBLE);
         txtV.startAnimation(fadeOutAnimation);


        }

        @Override
        public void onAccuracyChanged(Sensor arg0, int arg1) {
                // TODO Auto-generated method stub

        }

        @Override
        public void onSensorChanged(SensorEvent event) {
                // TODO Auto-generated method stub

                gx = event.values[0];
                gy = event.values[1];
                gz = event.values[2];
                g=Math.sqrt(gx*gx+gy*gy+gz*gz);
                fixsen=(Math.asin(gz/g))*180/Math.PI;
                fizsen=(Math.asin(gx/g))*180/Math.PI;

        }

    protected void onResume() {
        super.onResume();
        mSensorManager.registerListener(this, mGravity,
SensorManager.SENSOR_DELAY_NORMAL);
    }

    protected void onPause() {
        super.onPause();
        mSensorManager.unregisterListener(this);

    }

    public void lanzarSalir(View view){
```

```
        //Para salir de la aplicación
                Intent i = new Intent();
                setResult(RESULT_OK,i);
                finish();

       }



    private class Task extends AsyncTask <byte[], Void, Mat>{
        //Para poder hacer que aparezca un mensaje de espera por pantalla mientras se
realiza algún cálculo, hay que ponerlo todo dentro de una clase que extienda de
AsyncTask

         protected void onPreExecute() {
                dialog.show();
          }

         @Override
         protected Mat doInBackground(byte[]... data) {
                //Hace toda la segmentación
                Bitmap bitmap = BitmapFactory.decodeByteArray(d, 0,d.length);

                Mat img=new Mat();
                 Mat suref=new Mat();
                 String con_marcador=new String();
                 String dilatacion=new String();
                 String imagenYdil=new String();
                 String surefor=new String();
                 String imagenYssurefor=new String();
                 String mascara=new String();
                 String imagenYmascara=new String();
                 String grabblack=new String();
                 String original=new String();
                 String grabrrgb=new String();
                 int dilat=0;

                 Utils.bitmapToMat(bitmap, mRgba);
                 Imgproc.cvtColor(mRgba,mRgba,Imgproc.COLOR_RGB2BGR,3);
                 Core.transpose(mRgba, mRgba);

                 Core.flip(mRgba, mRgba, 180);
                 String path;

                 Camera.Parameters parameters=mCamera.getParameters();

                 if(tipoimagen==0){

                        try{
                        img = Utils.loadResource(CameraMain.this,
R.drawable.sil1_m,Highgui.CV_LOAD_IMAGE_COLOR);
                        }catch(IOException e){
                               Log.d(TAG, "Error prueba: " + e.getMessage());
                        }

                        try{
                        suref = Utils.loadResource(CameraMain.this,
R.drawable.sureforfrentefinal,Highgui.CV_LOAD_IMAGE_COLOR);
                        }catch(IOException e){
                               Log.d(TAG, "Error prueba: " + e.getMessage());
                        }

                        con_marcador="frente_con_marcador.jpg";
                        dilatacion="frente_dilatacion.jpg";
                        imagenYdil="frente_imageYdil.jpg";
                        surefor="frente_surefor.jpg";
                        imagenYssurefor="frente_imagenYsurefor.jpg";
                        mascara="frente_mascara.jpg";
                        imagenYmascara="frente_imageYmascara.jpg";
                        grabblack="frente_grabblack.jpg";
                        original="frente_original.jpg";
                        grabrrgb="frente_grabrrgb.jpg";
                        dilat=60;

                        focalLength=parameters.getFocalLength();
                        verticalAngle=parameters.getVerticalViewAngle();
```

```
                        horizontalAngle=parameters. getHorizontalViewAngle();

                        df_frente="Distancia Focal: "+focalLength+" mm";
                        ah_frente="Angulo Horizontal: "+horizontalAngle+" grados";
                        av_frente="Angulo Vertical: "+verticalAngle+" grados";

                }

                if(tipoimagen==1){

                        try{
                        img = Utils.loadResource(CameraMain.this,
R.drawable.sil3_m,Highgui.CV_LOAD_IMAGE_COLOR);
                        }catch(IOException e){
                                Log.d(TAG, "Error prueba: " + e.getMessage());
                        }

                        try{
                        suref = Utils.loadResource(CameraMain.this,
R.drawable.sureforperfilfinal,Highgui.CV_LOAD_IMAGE_COLOR);
                        }catch(IOException e){
                                Log.d(TAG, "Error prueba: " + e.getMessage());
                        }

                        con_marcador="perfil_con_marcador.jpg";
                        dilatacion="perfil_dilatacion.jpg";
                        imagenYdil="perfil_imageYdil.jpg";
                        surefor="perfil_surefor.jpg";
                        imagenYssurefor="perfil_imagenYsurefor.jpg";
                        mascara="perfil_mascara.jpg";
                        imagenYmascara="perfil_imageYmascara.jpg";
                        grabblack="perfil_grabblack.jpg";
                        original="perfil_original.jpg";
                        grabrrgb="perfil_grabrrgb.jpg";
                        dilat=60;


                }

                Mat copia=new Mat(mRgba.size(),CvType.CV_8UC3,new Scalar(0,0,0));

                Imgproc.threshold( img, img, 0, 255,0);

                double oldW=mRgba.width();
                double oldH=mRgba.height();

                double newH=750;
                double newW=Math.round((oldW/oldH)*newH);

                org.opencv.core.Size siz=new org.opencv.core.Size(newW,newH);
                Imgproc.resize(mRgba, mRgba, siz, 0, 0, Imgproc.INTER_LINEAR);
                Imgproc.resize(img, img, mRgba.size(), 0, 0, Imgproc.INTER_LINEAR);
                Imgproc.resize(suref, suref, mRgba.size(), 0, 0, Imgproc.INTER_LINEAR);

                Log.d(TAG, "mRgba height: "+mRgba.height()+" mRgba width:
"+mRgba.width());

                Mat copia2= new Mat();
                Mat copia3= new Mat();
                Mat copiaRgb=new Mat();
                mRgba.copyTo(copiaRgb);

                mRgba.copyTo(copia2);
                img.copyTo(copia3);

               Imgproc.cvtColor(copia3,copia3,Imgproc.COLOR_BGR2GRAY);
               Imgproc.threshold( copia3, copia3, 0, 255,0);

               Mat hierarchy1 = new Mat();
                List<MatOfPoint> contours1 = new ArrayList<MatOfPoint>();

                Imgproc.findContours( copia3, contours1, hierarchy1, Imgproc.RETR_CCOMP,
Imgproc.CHAIN_APPROX_SIMPLE);

               Imgproc.drawContours(copia2, contours1, -1, new Scalar( 0,0,255), 3);

                path=dir + File.separator + con_marcador;
```

```
                Highgui.imwrite(path,copia2);

               Rect rect=new Rect();

               Mat mask = new Mat();
               Mat fgdModel=new Mat();
               Mat bgdModel=new Mat();
               Mat gry=new Mat();

               Imgproc.cvtColor(img,gry,Imgproc.COLOR_BGR2GRAY);
               Imgproc.threshold( gry, gry, 0,
255,Imgproc.THRESH_BINARY+Imgproc.THRESH_OTSU);

               Mat fg=new Mat();
               Imgproc.threshold(gry,fg, 0, 3, 0);


               Mat bg=new Mat();
               Imgproc.threshold(gry,bg, 0, 2,1);

               Mat dil=new Mat();
               org.opencv.core.Size  se=new org.opencv.core.Size(dilat,dilat);
               Imgproc.dilate(gry, dil, Imgproc.getStructuringElement(Imgproc.MORPH_RECT
,se));


                path=dir + File.separator + dilatacion;//Grabar la dilatacion
                Highgui.imwrite(path,dil);

               Mat copiadil=new Mat();
               Mat copiagrysdil=new Mat();

               mRgba.copyTo(copiadil);
               dil.copyTo(copiagrysdil);

               Mat hierarchy4 = new Mat();
                List<MatOfPoint> contours4 = new ArrayList<MatOfPoint>();

                Imgproc.findContours(copiagrysdil, contours4, hierarchy4,
Imgproc.RETR_CCOMP, Imgproc.CHAIN_APPROX_SIMPLE);

               Imgproc.drawContours(copiadil, contours4, -1, new Scalar( 0,0,255), 3);

                path=dir + File.separator + imagenYdil;
                Highgui.imwrite(path,copiadil);

               Imgproc.threshold(dil,dil, 0, 2,1);

               Mat grysfor=new Mat();

               Imgproc.cvtColor(suref,grysfor,Imgproc.COLOR_BGR2GRAY);
               Imgproc.threshold( grysfor, grysfor, 0, 255,0);


                path=dir + File.separator +surefor;
                Highgui.imwrite(path,grysfor);

               Mat copiafor=new Mat();
               Mat copiagrysdfor=new Mat();

               mRgba.copyTo(copiafor);
               grysfor.copyTo(copiagrysdfor);

               Mat hierarchy3 = new Mat();
                List<MatOfPoint> contours3 = new ArrayList<MatOfPoint>();

                Imgproc.findContours(copiagrysdfor, contours3, hierarchy3,
Imgproc.RETR_CCOMP, Imgproc.CHAIN_APPROX_SIMPLE);

               Imgproc.drawContours(copiafor, contours3, -1, new Scalar( 0,0,255), 3);

                path=dir + File.separator + imagenYssurefor;
                Highgui.imwrite(path,copiafor);

                Imgproc.threshold(grysfor,grysfor, 0, 2,0);

                Mat maskp=new Mat();
```

```
                Core.add(fg, bg, mask);

                Core.subtract(mask, dil, maskp);

                Core.subtract(maskp,grysfor,mask);

                Mat pfg=new Mat();


                Core.subtract(maskp,fg , pfg);
                Core.add(pfg, grysfor, pfg);

                Imgproc.threshold(pfg,pfg, 1, 255,0);



                 path=dir + File.separator +mascara;//Grabar la mascara final
                 Highgui.imwrite(path,pfg);

                Mat hierarchy2 = new Mat();
                 List<MatOfPoint> contours2 = new ArrayList<MatOfPoint>();

                 Imgproc.findContours( pfg, contours2, hierarchy2, Imgproc.RETR_CCOMP,
Imgproc.CHAIN_APPROX_SIMPLE);


                Imgproc.drawContours(copiaRgb, contours2, -1, new Scalar( 0,0,255), 3);

                 path=dir + File.separator +imagenYmascara;
                 Highgui.imwrite(path,copiaRgb);

                 Imgproc.grabCut(mRgba, mask, rect, bgdModel, fgdModel, 2,
Imgproc.GC_INIT_WITH_MASK);

                 Mat mask1=new Mat();
                 Mat mask2=new Mat();

                 Core.compare(mask, new Scalar(3), mask1, Core.CMP_EQ);
                 Core.compare(mask, new Scalar(1), mask2, Core.CMP_EQ);

                 Core.add(mask1,mask2,mask);


                 path=dir + File.separator +grabblack;
                 Highgui.imwrite(path,mask);

                 Imgproc.threshold(mask,mask, 0, 255,0);

                 Mat foreground=new Mat(copia.size(),CvType.CV_8UC3,new Scalar(0,0,0));

                 Mat hierarchy = new Mat();
                 contours = new ArrayList<MatOfPoint>();

                 Imgproc.findContours( mask, contours, hierarchy, Imgproc.RETR_CCOMP,
Imgproc.CHAIN_APPROX_SIMPLE);

                 mRgba.copyTo(foreground,mask);

                 double largest_area=0;
                int largest_contour_index=0;

                for( int j = 0; j< contours.size(); j++ ){ // recorre los contornos
                   Mat contour = contours.get(j);
                    double a=Imgproc.contourArea( contour,false);  //  encuentra el area
del cada contorno
                    if(a>largest_area){
                    largest_area=a;
                    largest_contour_index=j;                //almacena el indice del mayor
contorno

                    }

                   }

                matContour=contours.get(largest_contour_index);
```

```
                Imgproc.drawContours(mRgba, contours, largest_contour_index, new Scalar(
0,0,255), 3);

                Mat end=new Mat();
                mRgba.copyTo(end);


                 path=dir + File.separator + original;
                 Highgui.imwrite(path,end);


                 path=dir + File.separator +grabrrgb;
                 Highgui.imwrite(path,foreground);

                 copia.release();
                 img.release();
                 copia2.release();
                 copia3.release();
                 copiaRgb.release();
                 hierarchy1.release();
                 mask.release();
                 gry.release();
                 dil.release();
                 grysfor.release();
                 hierarchy2.release();
                 hierarchy.release();
                 mask1.release();
                 mask2.release();
                 foreground.release();

                 return end;

        }

        @Override
        protected void onPostExecute(Mat m) {
                //Muestra por patalla el resultado de la segmentación y pregunta al
usuario si el resultado es correcto o no. Los distintos elementos se muestran tras una
pequeña animación. Se van haciendo visibles poco a poco.
                Imgproc.cvtColor(m,m,Imgproc.COLOR_BGR2RGB,3);
                 Bitmap bmpT = Bitmap.createBitmap(m.cols(), m.rows(),
Bitmap.Config.ARGB_8888);
                 Utils.matToBitmap(m, bmpT,true);

                 dialog.dismiss();
                 mPreview.setOnTouchListener(null);

                 Animation fadeInAnimation = AnimationUtils.loadAnimation(CameraMain.this,
R.anim.fade_in);


                 imgView.setImageBitmap(bmpT);
                 imgView.startAnimation(fadeInAnimation );


                 fadeInAnimation.setDuration(2000);
                 txtV.setVisibility(View.VISIBLE);
                 txtV.startAnimation(fadeInAnimation);
                 fadeInAnimation.setDuration(3000);
                 btnYes.setVisibility(View.VISIBLE);
                 btnYes.startAnimation(fadeInAnimation);
                 fadeInAnimation.setDuration(4000);
                 btnNo.setVisibility(View.VISIBLE);
                 btnNo.startAnimation(fadeInAnimation );

                 btnYes.setEnabled(true);
                 btnNo.setEnabled(true);

                    btnYes.setOnClickListener(new OnClickListener() {

                     public void onClick(View view) {

                        try {
                                        FileOutputStream fos = new
FileOutputStream(mPictureFileName);
```

```
                                Bitmap btmap = BitmapFactory.decodeByteArray(d,
0,d.length);

                                btmap.compress(Bitmap.CompressFormat.JPEG, 90, fos);


                            } catch (java.io.IOException e) {
                                Log.e("PictureDemo", "Exception in photoCallback", e);
                            }

                     grabarResultados();
                     String s=mPictureFileName+" saved";
                     showMessage(s);
                     Disconnect();
                    mCamera.startPreview();

                    if(tipoimagen==0){
                           mPreview.Imagen(tipoimagen);
                           imgView.setImageResource(R.drawable.sil3_t);
                           tipoimagen=1;
                            mPreview.setOnTouchListener(CameraMain.this);


                        }else if(tipoimagen==1){
                           mPreview.Imagen(tipoimagen);
                           tipoimagen=0;
                          mCamera.stopPreview();

                          setContentView(R.layout.resultados);

                          TxtFxF=(TextView)findViewById(R.id.txtFxF);
                          TxtFzF=(TextView)findViewById(R.id.txtFzF);
                          TxtDF=(TextView)findViewById(R.id.txtDF);
                          TxtAh=(TextView)findViewById(R.id.txtAh);
                          TxtAv=(TextView)findViewById(R.id.txtAv);
                          TxtFxP=(TextView)findViewById(R.id.txtFxP);
                          TxtFzP=(TextView)findViewById(R.id.txtFzP);
                          TxtE01f=(TextView)findViewById(R.id.txtE12f);
                          TxtE10f=(TextView)findViewById(R.id.txtE21f);
                          TxtE12f=(TextView)findViewById(R.id.txtE23f);
                          TxtE21f=(TextView)findViewById(R.id.txtE32f);
                          TxtE01p=(TextView)findViewById(R.id.txtE12p);
                          TxtE10p=(TextView)findViewById(R.id.txtE21p);
                          TxtE12p=(TextView)findViewById(R.id.txtE23p);
                          TxtE21p=(TextView)findViewById(R.id.txtE32p);
                          salir=(Button)findViewById(R.id.btnSalir);

                          TxtDF.setText(df_frente);
                          TxtAh.setText(ah_frente);
                          TxtAv.setText(av_frente);

                          TxtFxF.setText(fix_frente);
                          TxtFzF.setText(fiz_frente);
                          TxtFxP.setText(fix_perfil);
                          TxtFzP.setText(fiz_perfil);

                          TxtE01f.setText(""+e01_frente);
                          TxtE10f.setText(""+e10_frente);
                          TxtE12f.setText(""+e12_frente);
                          TxtE21f.setText(""+e21_frente);

                          TxtE01p.setText(""+e01_perfil);
                          TxtE10p.setText(""+e10_perfil);
                          TxtE12p.setText(""+e12_perfil);
                          TxtE21p.setText(""+e21_perfil);




                       }
                    }
                });


                btnNo.setOnClickListener(new OnClickListener() {
```

```
                public void onClick(View view) {
                  String s=mPictureFileName+" not saved";
                  showMessage(s);
                  Disconnect();

                  mCamera.startPreview();

                              if(tipoimagen==0){
                              mPreview.Imagen(tipoimagen);
                              imgView.setImageResource(R.drawable.sil1_t);
                              mPreview.setOnTouchListener(CameraMain.this);


                             }else if(tipoimagen==1){
                              mPreview.Imagen(tipoimagen);
                              imgView.setImageResource(R.drawable.sil3_t);
                              mPreview.setOnTouchListener(CameraMain.this);

                           }
                        }


                });

        }


    @Override
    protected void onProgressUpdate(Void... values) {

    }
}

//A PARTIR DE AQUI ESTAS FUNCIONES SE USAN PARA GUARDAR RESULTADOS

public void grabarResultados(){

   //Escribe un fichero xml en /sdcard/Android/data/org.example.proyectofinal/files

   String fichero="resultados.xml";
   Element var;
   Bundle extras = getIntent().getExtras();
   double a=extras.getDouble("altura");
   double p=extras.getDouble("peso");
   int ed=extras.getInt("edad");
   Mat rotacion=new Mat(3,3,CvType.CV_64FC1,new Scalar(1));
   rotacion.put(2, 0, 0);
   rotacion.put(0, 2, 0);
   String rot=new String();


        if(tipoimagen==0){
               valores[0]=focalLength;
               valores[1]=horizontalAngle;
               valores[2]=verticalAngle;
               valores[3]=fix;
               valores[4]=fiz;
               valores[5]=Gx;
               valores[6]=Gy;
               valores[7]=Gz;
               tag="Contorno_Matriz_Frente";
               names=nfrente;
               alt=1;

               rotacion.put(0, 1, e01_frente);
               rotacion.put(1, 0, e10_frente);
               rotacion.put(1, 2, e12_frente);
               rotacion.put(2, 1, e21_frente);
               rot="Matriz_Rotacion_Frente";
        }else if(tipoimagen==1){

               valores[0]=fix;
               valores[1]=fiz;
               valores[2]=Gx;
               valores[3]=Gy;
```

```
                        valores[4]=Gz;
                        tag="Contorno_Matriz_Perfil";
                        names=nperfil;
                        alt=2;

                        rotacion.put(0, 1, e01_perfil);
                        rotacion.put(1, 0, e10_perfil);
                        rotacion.put(1, 2, e12_perfil);
                        rotacion.put(2, 1, e21_perfil);
                        rot="Matriz_Rotacion_Perfil";
                }


        try{
                File file=new File(myDir,fichero);
                if( file.isFile() == false ){
                doc =
DocumentBuilderFactory.newInstance().newDocumentBuilder().newDocument();

             rootElement = doc.createElement("opencv_storage");
             doc.appendChild(rootElement);
                }else{
                 doc =
DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(file);
                 rootElement = doc.getDocumentElement();
                }


                if(alt==1){

                        var=doc.createElement("Altura");
                 var.appendChild( doc.createTextNode(String.valueOf(a)));
                 rootElement.appendChild(var);

                        var=doc.createElement("Peso");
                 var.appendChild( doc.createTextNode(String.valueOf(p)));
                 rootElement.appendChild(var);

                        var=doc.createElement("Edad");
                 var.appendChild( doc.createTextNode(String.valueOf(ed)));
                 rootElement.appendChild(var);

                        var = doc.createElement("Piture_Height_Frente");
                        var.appendChild( doc.createTextNode(
String.valueOf(mRgba.height())));
                        rootElement.appendChild(var);

                        var = doc.createElement("Piture_Width_Frente");
                        var.appendChild( doc.createTextNode(
String.valueOf(mRgba.width())));
                        rootElement.appendChild(var);

                        Log.d(TAG, "save Picture height: "+mRgba.height()+" Picture width:
"+mRgba.width());
                }else{

                        var = doc.createElement("Piture_Height_Perfil");
                        var.appendChild( doc.createTextNode(
String.valueOf(mRgba.height())));
                        rootElement.appendChild(var);

                        var = doc.createElement("Piture_Width_Perfil");
                        var.appendChild( doc.createTextNode(
String.valueOf(mRgba.width())));
                        rootElement.appendChild(var);
                }



             for(int k=0;k<names.length;k++){
                 var = doc.createElement(names[k]);
                 var.appendChild( doc.createTextNode( String.valueOf(valores[k])));
                 rootElement.appendChild(var);
             }
```

```
            writeMat(rot,rotacion,0);
               writeMat(tag,matContour,1);

            DOMSource source = new DOMSource(doc);

            StreamResult result = new StreamResult(file);

            // write to xml file
            Transformer transformer = TransformerFactory.newInstance().newTransformer();
            transformer.setOutputProperty(OutputKeys.INDENT, "yes");

            // do it
            transformer.transform(source, result);

        } catch(Exception e) {
            e.printStackTrace();
        }


    }

    public void writeMat(String tag, Mat mat, int tm) {
       //Método que se encarga de almacenar las matrices en el fichero xml
        try {

            Element matrix = doc.createElement(tag);
            matrix.setAttribute("type_id", "opencv-matrix");
            rootElement.appendChild(matrix);

            Element rows = doc.createElement("rows");
            rows.appendChild( doc.createTextNode( String.valueOf(mat.rows()) ));
            Element cols = doc.createElement("cols");
            if (tm==1){
               cols.appendChild( doc.createTextNode( String.valueOf(mat.channels()) ));
            }else{
               cols.appendChild( doc.createTextNode( String.valueOf(mat.cols()) ));
            }
            Element dt = doc.createElement("dt");
            String dtStr;
            int type = mat.type();
            if(type == CvType.CV_64FC1 ) { // type == CvType.CV_64FC1 Tipo de dato de la
matriz de rotación
                dtStr = "d";
            }
            else if( type == CvType.CV_32SC2 ) { // type == CvType.CV_32SC1 el tipo de
dato de la matriz que almacena el contorno
                dtStr = "i";
            }

            else {
                dtStr = "unknown";
            }
            dt.appendChild( doc.createTextNode( dtStr ));

            Element data = doc.createElement("data");
            String dataStr = dataStringBuilder( mat );
            data.appendChild( doc.createTextNode( dataStr ));

            // append all to matrix
            matrix.appendChild( rows );
            matrix.appendChild( cols );
            matrix.appendChild( dt );
            matrix.appendChild( data );

        } catch(Exception e) {
            e.printStackTrace();
        }
    }

    private String dataStringBuilder(Mat mat) {
        StringBuilder sb = new StringBuilder();
        int rows = mat.rows();
        int cols = mat.cols();
        int type = mat.type();
```

```
        if( type == CvType.CV_64FC1 ) {                                    //Para montar
la matriz en el fichero xml
            double fs[] = new double[1];
            for( int r=0 ; r<rows ; r++ ) {
                for( int c=0 ; c<cols ; c++ ) {
                    mat.get(r, c, fs);
                    sb.append( String.valueOf(fs[0]));
                    sb.append( ' ' );
                }
                sb.append( '\n' );
            }
        }
        else if( type == CvType.CV_32SC2 ) {               //Para montar la matriz en el
fichero xml
            int is[] = new int[2];
            for( int r=0 ; r<rows ; r++ ) {
                for( int c=0 ; c<cols ; c++ ) {
                    mat.get(r, c, is);
                    sb.append( String.valueOf(is[0]));
                    sb.append( ' ' );
                    sb.append( String.valueOf(is[1]));
                }
                sb.append( '\n' );
            }
        }

        else {
            sb.append("unknown type\n");
        }

        return sb.toString();
    }


}
```

Preview.java

Clase que se encarga de configurar la cámara del dispositivo y que lo que esta capte se muestre de forma correcta por la pantalla.

```
package org.example.proyectofinal;


import java.io.IOException;
import java.util.List;
import android.annotation.SuppressLint;
import android.annotation.TargetApi;
import android.content.Context;
import android.graphics.ImageFormat;
import android.hardware.Camera;
import android.hardware.Camera.Parameters;
import android.hardware.Camera.Size;
import android.os.Build;
import android.util.Log;
import android.view.Surface;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.ImageView;

@SuppressLint("ViewConstructor")
public class Preview extends SurfaceView  implements
SurfaceHolder.Callback,Camera.PreviewCallback{

    SurfaceView mSurfaceView;
       private SurfaceHolder mHolder;
    Camera mCamera;
    List<Size> mSupportedPreviewSizes;
    Camera.Size mPreviewSize;
    private Context mContext;
    protected List<Camera.Size> mPreviewSizeList;
    protected List<Camera.Size> mPictureSizeList;
       int tipoimagen;
```

```
        ImageView imgView;
        Button btnYes;
        Button btnNo;
        boolean confirm;
        private int index = 0;
        private int maxPicw, maxPich;



    private static final String TAG ="Mas Pruebas";

   public Preview(Context context,Camera camera) {
        super(context);
        mContext=context;

        mCamera=camera;
        if (mCamera != null) {
            mSupportedPreviewSizes = mCamera.getParameters().getSupportedPreviewSizes();
            requestLayout();
        }
        // Install a SurfaceHolder.Callback so we get notified when the
        // underlying surface is created and destroyed.
        mHolder = getHolder();
        mHolder.addCallback(this);
        if(Build.VERSION.SDK_INT < Build.VERSION_CODES.HONEYCOMB) //Sólo funciona para
versiones anteriores a la 3.0
              mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);


    }



    public void surfaceDestroyed(SurfaceHolder holder) {
        // Surface will be destroyed when we return, so stop the preview.
        if (mCamera != null) {
            // Call stopPreview() to stop updating the preview surface.
            mCamera.stopPreview();
            mCamera.release();
            mCamera=null;
        }
    }

    /**
     * When this function returns, mCamera will be null.
     */

        @Override
        public void surfaceCreated(SurfaceHolder holder) {
               // TODO Auto-generated method stub


        }

        @Override
        protected void onLayout(boolean changed, int l, int t, int r, int b) {
               // TODO Auto-generated method stub

        }


        @Override
        public void surfaceChanged(SurfaceHolder holder, int format, int width,int
height) {
               // TODO Auto-generated method stub

        // start preview with new settings

     try {
            mCamera.stopPreview();
        } catch (Exception e) {
            e.printStackTrace();
            // ignore: tried to stop a non-existent preview
        }
```

```
              Camera.Parameters parameters=mCamera.getParameters();
              configureCameraParameters(parameters);

        List<Size> sizes = parameters.getSupportedPreviewSizes();
        Size optimalSize = getOptimalPreviewSize(sizes,width, height);
        Log.d(TAG, "Surface size is " + width + "w " + height + "h");
        Log.d(TAG, "Optimal size is " + optimalSize.width + "w " + optimalSize.height +
"h");
        parameters.setPreviewSize(optimalSize .width,optimalSize.height);
        requestLayout();

        parameters.setFlashMode(Parameters.FLASH_MODE_ON);
        parameters.setFocusMode(Parameters.FOCUS_MODE_AUTO);
              parameters.setJpegQuality(100);
              Log.d(TAG, "Jpeg quality after: " +parameters.getJpegQuality());
        parameters.setPictureFormat(ImageFormat.JPEG);

        setMaxPicturesize(parameters);

        mCamera.setParameters(parameters);



            try   {

                mCamera.setPreviewDisplay(mHolder);
                mCamera.startPreview();

             } catch (IOException e) {

             }


                    }


    @TargetApi(Build.VERSION_CODES.GINGERBREAD)            //lo pide añadir eclipse
porque la aplicacion funciona desde la version 8 de android hasta la 18
       @SuppressLint("NewApi")
       protected void configureCameraParameters(Camera.Parameters cameraParams) {
       //Para cambiar la rotacion de la camara
        android.hardware.Camera.CameraInfo info = new
android.hardware.Camera.CameraInfo();
        android.hardware.Camera.getCameraInfo(0, info);
        int rotation = ((WindowManager) mContext
                        .getSystemService(Context.WINDOW_SERVICE)).getDefaultDisplay()
                        .getRotation();
        int degrees = 0, result = 0;

        switch (rotation) {
        case Surface.ROTATION_0:
                degrees = 0;
                break;
        case Surface.ROTATION_90:
                degrees = 90;
                break;
        case Surface.ROTATION_180:
                degrees = 180;
                break;
        case Surface.ROTATION_270:
                degrees = 270;
                break;
        }

        Log.v(TAG, "angle: " + degrees);
        result = (info.orientation - degrees + 360) % 360;
        mCamera.setDisplayOrientation(result);


    }

       private Camera.Size getOptimalPreviewSize(List<Camera.Size> sizes, int w, int h)
{
```

```
            //Selecciona el mejor tamaño del preview para mostrar en la superficie
creada para la pantalla.

        final double ASPECT_TOLERANCE = 0.1;
        double targetRatio=(double)h / w;

        if (sizes == null) return null;

        Camera.Size optimalSize = null;
        double minDiff = Double.MAX_VALUE;

        int targetHeight = h;

        for (Camera.Size size : sizes) {
            double ratio = (double) size.width / size.height;
            if (Math.abs(ratio - targetRatio) > ASPECT_TOLERANCE) continue;
            if (Math.abs(size.height - targetHeight) < minDiff) {
                optimalSize = size;
                minDiff = Math.abs(size.height - targetHeight);
            }
        }

        if (optimalSize == null) {
            minDiff = Double.MAX_VALUE;
            for (Camera.Size size : sizes) {
                if (Math.abs(size.height - targetHeight) < minDiff) {
                    optimalSize = size;
                    minDiff = Math.abs(size.height - targetHeight);
                }
            }
        }
        return optimalSize;
    }



    public void Imagen(int tipoimg){
        tipoimagen=tipoimg;
    }


        @Override
        public void onPreviewFrame(byte[] arg0, Camera arg1) {
               // TODO Auto-generated method stub

        }

    public void setMaxPicturesize(Camera.Parameters cameraParams){
         //Para seleccionar el maximo tamañoo de foto, aunque por defecto ya va
seleccionada, pero mejor hacerlo por si acaso
        List<Camera.Size> sizesPic = cameraParams.getSupportedPictureSizes();
        //Toast.makeText(mContext,"Supported Sizes: " +
sizesPic,Toast.LENGTH_LONG).show();
        boolean cond=false;

        for (int i = 0; i < sizesPic.size(); i++){
            Size s = sizesPic.get(i);

            float size = (s.height * s.width)/1000000;

            if (size > 2.9 && size < 3.9) {
                index = i;
                Log.d(TAG, "3MPx");
                cond=true;
            }
        }

        if(cond==false){
               index=0;
        }

        maxPicw=sizesPic.get(index).width;
        maxPich=sizesPic.get(index).height;
        Log.d(TAG, "PicSize:  " + maxPicw + " w x" + maxPich + "h");
        cameraParams.setPictureSize(maxPicw, maxPich);

        mCamera.setParameters(cameraParams);
```

```
    }
}
```

9.2. Anexo B: Layouts y ficheros de configuración

splash_screen.xml

Este layout simplemente cargará una imagen cuando se llame desde la clase SplashScreenActivity.java.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_height="fill_parent"
            android:layout_width="fill_parent"
            android:orientation="vertical"
            android:background="@drawable/fondo">

    <ImageView
        android:id="@+id/sampleImageView"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_gravity="center_vertical|center_horizontal"
        android:gravity="center"
        android:src="@drawable/til"
        android:contentDescription="@string/desc"/>

</LinearLayout>
```

activity_main.xml

Layout que se encarga de mostrar por pantalla el tutorial mediante el componente ViewFlipper, al que hay que pasarle una serie de ficheros xml para ir pasando hacia adelante o hacia atrás de las vistas con solo deslizar el dedo por la pantalla del smartphone. Las vistas por las que no vamos moviendo en la aplicación se muestran a continuación de activity_main.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/RelativeLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:background="@drawable/fondo"
    >

    <ViewFlipper
        android:id="@+id/viewFlipper"
        android:layout_width="fill_parent"
        android:layout_height="500dp"
        android:layout_above="@+id/progressBar1"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true" >


        <include layout="@layout/dos" />

        <include layout="@layout/tres" />

        <include layout="@layout/cuatro" />

        <include layout="@layout/cinco" />

        <include layout="@layout/seis" />

        <include layout="@layout/siete" />

        <include layout="@layout/ocho" />

    </ViewFlipper>

    <ProgressBar
        android:id="@+id/progressBar1"
        style="?android:attr/progressBarStyleHorizontal"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginLeft="80dp"
        android:max="6"
        />

    <TextView
        android:id="@+id/txtView"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_alignBottom="@id/progressBar1"
        android:layout_alignLeft="@id/progressBar1"
        android:layout_alignRight="@id/progressBar1"
        android:layout_alignTop="@id/progressBar1"
        android:background="#00000000"
        android:gravity="center"
        android:textColor="#FFFFFF" />

</RelativeLayout>
```

dos.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/RelativeLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
              style="@style/Appstyle"
        android:layout_alignParentTop="true"/>

    <TextView
        android:id="@+id/TextView2"
              style="@style/Appstyle.pasos"
        android:text="@string/info"/>

</RelativeLayout>
```

tres.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center_horizontal"
    android:orientation="vertical" >

    <TextView
        style="@style/Appstyle"
        android:layout_alignParentTop="true"
        android:gravity="center|center_vertical"
        android:text="@string/paso_uno"/>

    <LinearLayout
        android:id="@+id/LinearLayout1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:orientation="vertical" >

        <TextView
            style="@style/Appstyle.pasos"
            android:drawableTop="@drawable/frente_1"
            android:text="@string/pasouno" />

    </LinearLayout>

</RelativeLayout>
```

cuatro.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/RelativeLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        style="@style/Appstyle"
        android:layout_alignParentTop="true"
        android:text="@string/paso_dos"/>

    <TextView
        style="@style/Appstyle.pasos"
        android:drawableTop="@drawable/frente_con_marcador_1"
        android:text="@string/pasodos" />

</RelativeLayout>
```

cinco.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/RelativeLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
              style="@style/Appstyle"
        android:layout_alignParentTop="true"
        android:text="@string/paso_tres"/>

    <TextView
        android:id="@+id/txtPasotres"
        style="@style/Appstyle.pasos"/>

</RelativeLayout>
```

seis.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/RelativeLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
              style="@style/Appstyle"
        android:layout_alignParentTop="true"
        android:text="@string/paso_cuatro"/>

    <TextView
              android:id="@+id/txtPasocuatro"
              style="@style/Appstyle.pasos"
        android:drawableTop="@drawable/perfil_con_marcador_1"/>

</RelativeLayout>
```

siete.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
         style="@style/Appstyle"
        android:layout_alignParentTop="true"
        android:text="@string/paso_cinco"/>

       <LinearLayout
```

```
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:orientation="vertical" >

              <TextView
                style="@style/Appstyle.pasos"
               android:id="@+id/txtPasocinco"
               />

       </LinearLayout>
</RelativeLayout>
```

ocho.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
              style="@style/Appstyle"
              android:text="@string/paso_seis"/>

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical|center_horizontal"
        android:layout_marginTop="5dp"
        android:gravity="center"
        android:text="@string/datos"
               android:textColor="#FFFFFF"
        android:textSize="20sp" />

       <EditText
           android:id="@+id/TxtAltura"
           android:layout_width="fill_parent"
           android:layout_height="wrap_content"
           android:hint="@string/altura"
           android:inputType="numberDecimal"
           android:layout_marginTop="5dp"/>


       <EditText
        android:id="@+id/TxtEdad"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/edad"
        android:inputType="number"
        android:layout_marginTop="5dp"/>


       <EditText
           android:id="@+id/TxtPeso"
           android:layout_width="fill_parent"
           android:layout_height="wrap_content"
           android:hint="@string/peso"
           android:inputType="numberDecimal"
           android:layout_marginTop="5dp"/>

       <Button
           android:id="@+id/btnLanzar"
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:layout_marginTop="20dp"
           android:onClick="lanzarEmpezar"
           android:text="@string/lanzar"
           android:textSize="20sp" />

</LinearLayout>
```

camera.xml

Layout que se encarga de mostrar por pantalla lo que capta la cámara y de mostrar los mensajes de espera, los resultados de la segmentación y de preguntar si estos son correctos o no.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/RelativeLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <FrameLayout
        android:id="@+id/preview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:orientation="vertical" >
    </FrameLayout>


    <ImageView
       android:id="@+id/imgView"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent"
       android:layout_centerHorizontal="true"
       android:layout_centerVertical="true"
       android:scaleType="centerCrop"
       android:contentDescription="@string/desc"/>

    <Button
        android:id="@+id/btnSi"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:onClick="lanzarEmpezar"
        android:text="@string/si"
        android:textSize="20sp"
        android:visibility="invisible" />

       <Button
           android:id="@+id/btnNon"
           android:layout_width="150dp"
           android:layout_height="wrap_content"
           android:layout_alignParentBottom="true"
           android:layout_alignParentRight="true"
           android:layout_centerInParent="false"
           android:onClick="lanzarEmpezar"
           android:text="@string/no"
           android:textSize="20sp"
           android:visibility="invisible" />

       <TextView
           android:id="@+id/txtView3"
           android:layout_width="fill_parent"
           android:layout_height="wrap_content"
           android:layout_above="@+id/btnSi"
           android:layout_alignParentLeft="true"
           android:gravity="center"
           android:text="@string/pregunta"
           android:textColor="#FFFFFF"
           android:textSize="23sp"
           android:visibility="invisible" />

</RelativeLayout>
```

resultados.xml

Este layout muestra al final de la aplicación los parámetros de la cámara, los ángulos de inclinación de esta al tomar las fotografías y las matrices de rotación de cada fotografía. Se utiliza para comprobar que los datos salen de forma correcta, no tendría que ir en la aplicación final que se lanzará al usuario medio.

```
<RelativeLayout xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/RelativeLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:orientation="vertical" >


    <TextView
        android:id="@+id/txtRes"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:gravity="center"
        android:text="@string/resultados"
        android:textColor="#FFFFFF"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/txtPC"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/txtRes"
        android:gravity="center"
        android:paddingTop="5dp"
        android:text="Parámetros de la cámara:"
        android:textColor="#FFFFFF" />

    <TextView
        android:id="@+id/txtDF"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/txtPC"
        android:gravity="center"
        android:paddingTop="3dp"
        android:textColor="#FFFFFF" />

    <TextView
        android:id="@+id/txtAh"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/txtDF"
        android:gravity="center"
        android:textColor="#FFFFFF" />

    <TextView
        android:id="@+id/txtAv"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/txtAh"
        android:gravity="center"
        android:textColor="#FFFFFF" />

    <TextView
        android:id="@+id/txtIFF"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/txtAv"
        android:gravity="center"
        android:paddingTop="5dp"
        android:text="Inclinación foto frente:"
        android:textColor="#FFFFFF" />

    <TextView
        android:id="@+id/txtFxF"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
```

```
        android:layout_below="@+id/txtIFF"
        android:gravity="center"
        android:paddingTop="3dp"
        android:textColor="#FFFFFF" />

    <TextView
        android:id="@+id/txtFzF"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/txtFxF"
        android:gravity="center"
        android:textColor="#FFFFFF" />

    <TextView
        android:id="@+id/txtIFP"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/txtFzF"
        android:gravity="center"
        android:paddingTop="5dp"
        android:text="Inclinación foto perfil:"
        android:textColor="#FFFFFF" />

    <TextView
        android:id="@+id/txtFxP"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/txtIFP"
        android:gravity="center"
        android:paddingTop="3dp"
        android:textColor="#FFFFFF" />

    <TextView
        android:id="@+id/txtFzP"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/txtFxP"
        android:gravity="center"
        android:textColor="#FFFFFF" />

    <TextView
        android:id="@+id/txtMRF"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/txtFzP"
        android:gravity="center"
        android:paddingTop="5dp"
        android:text="Matriz de rotación frente:"
        android:textColor="#FFFFFF" />

    <TableLayout
        android:id="@+id/TableLayout1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/txtMRF" >

        <TableRow android:gravity="center" >

            <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:text="1"
                android:textColor="#FFFFFF" />

            <TextView
                android:id="@+id/txtE12f"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:textColor="#FFFFFF" />
```

```
            <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:text="0"
                android:textColor="#FFFFFF" />
        </TableRow>

        <TableRow android:gravity="center" >

            <TextView
                android:id="@+id/txtE21f"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:textColor="#FFFFFF" />

            <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:text="1"
                android:textColor="#FFFFFF" />

            <TextView
                android:id="@+id/txtE23f"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:textColor="#FFFFFF" />
        </TableRow>

        <TableRow android:gravity="center" >

            <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:text="0"
                android:textColor="#FFFFFF" />

            <TextView
                android:id="@+id/txtE32f"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:textColor="#FFFFFF" />

            <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:text="1"
                android:textColor="#FFFFFF" />
        </TableRow>
    </TableLayout>

    <TextView
        android:id="@+id/txtMRP"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/TableLayout1"
        android:gravity="center"
        android:paddingTop="5dp"
        android:text="Matriz de rotación perfil:"
        android:textColor="#FFFFFF" />

    <TableLayout
        android:id="@+id/TableLayout2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/txtMRP" >

        <TableRow android:gravity="center" >
```

```
            <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:text="1"
                android:textColor="#FFFFFF" />

            <TextView
                android:id="@+id/txtE12p"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:textColor="#FFFFFF" />

            <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:text="0"
                android:textColor="#FFFFFF" />
        </TableRow>

        <TableRow android:gravity="center" >

            <TextView
                android:id="@+id/txtE21p"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:textColor="#FFFFFF" />

            <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:text="1"
                android:textColor="#FFFFFF" />

            <TextView
                android:id="@+id/txtE23p"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:textColor="#FFFFFF" />
        </TableRow>

        <TableRow android:gravity="center" >

            <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:text="0"
                android:textColor="#FFFFFF" />

            <TextView
                android:id="@+id/txtE32p"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:textColor="#FFFFFF" />

            <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:text="1"
                android:textColor="#FFFFFF" />
        </TableRow>
    </TableLayout>

    <Button
        android:id="@+id/btnSalir"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
```

```
        android:layout_below="@+id/TableLayout2"
        android:layout_marginTop="20dp"
        android:onClick="lanzarSalir"
        android:text="Salir"
        android:textSize="20sp" />

</RelativeLayout>
```

strings.xml

Fichero que contiene todos los valores de tipo string que aparecen en los layouts para poder cambiar su valor sin necesidad de ir al código fuente.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

        <string name="app_name">ProyectoFinal</string>
        <string name="lanzar">¡Empezar!</string>
        <string name="info">Bienvenido. Con esta aplicación podrás obtener, a partir de
tu silueta, un modelo en 3D de tu cuerpo para compras online. Sólo tienes que seguir
unos sencillos pasos que se enumeran a continuación.</string>
        <string name="altura">Altura (en metros)</string>
        <string name="datos">Para finalizar le pedimos que introduzca una serie de datos
adicionales para poder obtener su silueta en 3D</string>
    <string name="edad">Edad</string>
        <string name="peso">Peso (en kg)</string>
        <string name="pregunta">¿Es el resultado correcto?</string>
        <string name="si">SI</string>
        <string name="no">NO</string>
        <string name="pasouno"> Usar ropa ajustada y oscura, como se ve en la imagen. Las
fotografías tienen que tomarse delante de una pared blanca y sin objetos delante y
detrás del usuario  </string>
        <string name="pasodos"> Primero se hará una foto de frente. Intenta que la
persona encaje lo mejor posible en la silueta que aparecerá en pantalla, como se muestra
en la imagen</string>
        <string name="pasotres"> Para hacer la foto basta con tocar la pantalla y después
de ajustarse solo el autofoco se tomara la foto y se mostrará un mensaje de
espera</string>
        <string name="pasotres_uno"> Tras el mensaje se mostrará el resultado, que tiene
que ser como el de la imagen de arriba, marcándose el contorno\n</string>
        <string name="pasotres_dos"> Cuando se muestra el resultado siempre se pregunta
si este es correcto, pudiendo repetirse por si algo sale mal, como en la foto
superior</string>
        <string name="pasocuatro"> Luego se hará otra foto de perfil en el lugar elegido.
Cuando la persona encaje lo mejor posible en la silueta que aparecerá en pantalla se
puede tomar la foto</string>
        <string name="pasocuatro_uno"> Los brazos tienen que estar lo más pegados
posibles al cuerpo para que no deformen la silueta,como se ve en la imagen\n</string>
        <string name="pasocinco"> Después del mensaje de espera se mostrará el resultado
por pantalla, que tiene que ser como la imagen superior, marcándose el contorno
</string>
        <string name="pasocinco_uno"> Al igual que antes se pregunta si el resultado es
correcto, pudiendo repetirse por si algo sale mal, como podemos ver en la imagen
superior</string>
        <string name="pasocinco_dos"> Una vez acabado el proceso ya se pueden enviar los
contornos obtenidos y los datos a rellenar, no se enviaran las fotos enteras, sólo los
datos de los contornos</string>
        <string name="paso_uno"> Paso 1</string>
        <string name="paso_dos"> Paso 2</string>
        <string name="paso_tres"> Paso 3</string>
        <string name="paso_cuatro"> Paso 4</string>
        <string name="paso_cinco"> Paso 5</string>
        <string name="paso_seis"> Paso 6</string>
        <string name="resultados"><u><b>Resultados</b></u></string>
        <string name="desc">Descriptor para las imágenes</string>
        <string name="action_settings">Elementos del menú</string>


</resources>
```

styles.xml

Fichero que recoge los estilos empleados en los diferentes textos mostrados en los layouts.

```
<resources>

    <!--
        Base application theme, dependent on API level. This theme is replaced
        by AppBaseTheme from res/values-vXX/styles.xml on newer devices.
    -->
    <style name="AppBaseTheme" parent="android:Theme.Light">
        <!--
            Theme customizations available in newer API levels can go in
            res/values-vXX/styles.xml, while customizations related to
            backward-compatibility can go here.
        -->
    </style>

    <!-- Application theme. -->
    <style name="AppTheme" parent="AppBaseTheme">
        <!-- All customizations that are NOT specific to a particular API-level can go
here. -->
    </style>


    <style name="Appstyle"
        parent="@android:style/TextAppearance.Medium">
                <item name="android:layout_width">fill_parent</item>
                 <item name="android:layout_height">wrap_content</item>
                 <item name="android:gravity">center</item>
                 <item name="android:textSize">23sp</item>
                 <item name="android:textColor">#FFFFFF</item>
     </style>


    <style name="Appstyle.pasos" >

              <item
        name="android:layout_gravity">center_vertical|center_horizontal</item>
              <item name="android:layout_centerVertical">true</item>
              <item name="android:layout_centerHorizontal">true</item>
              <item name="android:textSize">16sp</item>
              <item name="android:drawablePadding">10dp</item>


    </style>

</resources>
```

fade_in.xml

Animación muy simple para que el elemento seleccionado vaya apareciendo poco a poco por pantalla.

```
<?xml version="1.0" encoding="utf-8"?>
 <set xmlns:android="http://schemas.android.com/apk/res/android">
    <alpha android:fromAlpha="0.0" android:toAlpha="1.0"
     android:interpolator="@android:anim/accelerate_interpolator"
         android:duration="1000"
    />
 </set>
```

fade_out.xml

Animación muy simple para que el elemento seleccionado vaya desapareciendo poco a poco por pantalla.

```
<?xml version="1.0" encoding="utf-8"?>
 <set xmlns:android="http://schemas.android.com/apk/res/android">
    <alpha android:fromAlpha="1.0" android:toAlpha="0.0"
     android:interpolator="@android:anim/accelerate_interpolator"
     android:duration="1000"
    />
 </set>
```

slide_right.xml

Animación para entrada y salida de vistas hacia la derecha.

```
<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android"
android:interpolator="@android:anim/accelerate_interpolator">
    <translate android:fromXDelta="-100%p" android:toXDelta="0"
        android:duration="300" />
</set>
```

slide_left.xml

Animación para entrada y salida de vistas hacia la izquierda.

```
<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android"
android:interpolator="@android:anim/accelerate_interpolator">
    <translate android:fromXDelta="100%p" android:toXDelta="0"
        android:duration="300" />
</set>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.example.proyectofinal"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />

    <uses-permission android:name="android.permission.CAMERA" />
    <uses-feature android:name="android.hardware.camera" />
    <uses-feature android:name="android.hardware.camera.autofocus"/>
    <uses-feature android:name="android.hardware.camera.flash" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@android:style/Theme.NoTitleBar.Fullscreen" >
        <activity
            android:name="org.example.proyectofinal.SplashScreenActivity"
            android:label="@string/app_name"
            android:screenOrientation="portrait"
            android:configChanges="keyboardHidden|orientation">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="org.example.proyectofinal.MainActivity"
            android:label="@string/app_name"
            android:configChanges="keyboardHidden|orientation"
                      android:screenOrientation="portrait">

        </activity>
        <activity
            android:name="org.example.proyectofinal.CameraMain"
            android:label="@string/app_name"
            android:screenOrientation="portrait"
            android:configChanges="keyboardHidden|orientation">
        </activity>
    </application>
</manifest>
```

Almacenamiento de resultados

Ejemplo de cómo se almacenan los resultados de los contornos y los parámetros de la cámara en el fichero de resultados que tiene que ser en formato xml para poder leerlo en el servidor.

```
<?xml version="1.0" encoding="UTF-8"?><opencv_storage>
<Altura>1.73</Altura>
<Peso>65.3</Peso>
<Edad>32</Edad>
<Piture_Height_Frente>750</Piture_Height_Frente>
<Piture_Width_Frente>563</Piture_Width_Frente>
<Distancia_Focal>3.5399999618530273</Distancia_Focal>
<Angulo_Horizontal>66.4000015258789</Angulo_Horizontal>
<Angulo_Vertical>66.4000015258789</Angulo_Vertical>
<Giro_eje_x_Frente>-2.867</Giro_eje_x_Frente>
<Giro_eje_z_Frente>2.169</Giro_eje_z_Frente>
<gx_Frente>0.368</gx_Frente>
<gy_Frente>9.701</gy_Frente>
<gz_Frente>-0.486</gz_Frente>
<Matriz_Rotacion_Frente type_id="opencv-matrix">
<rows>3</rows>
<cols>3</cols>
<dt>d</dt>
<data>1.0 -0.038 0.0
0.038 1.0 0.05
0.0 -0.05 1.0
</data>
</Matriz_Rotacion_Frente>
<Contorno_Matriz_Frente type_id="opencv-matrix">
<rows>1148</rows>
<cols>2</cols>
<dt>i</dt>
<data> Aqui irían los puntos de las coordenadas que forman el contorno, que en este caso
serían 1148 filas de puntos para la foto de frente
</data>
</Contorno_Matriz_Frente>
<Piture_Height_Perfil>750</Piture_Height_Perfil>
<Piture_Width_Perfil>563</Piture_Width_Perfil>
<Giro_eje_x_Perfil>-5.0</Giro_eje_x_Perfil>
<Giro_eje_z_Perfil>-1.0</Giro_eje_z_Perfil>
<gx_Perfil>-0.17</gx_Perfil>
<gy_Perfil>9.795</gy_Perfil>
<gz_Perfil>-0.825</gz_Perfil>
<Matriz_Rotacion_Perfil type_id="opencv-matrix">
<rows>3</rows>
<cols>3</cols>
<dt>d</dt>
<data>1.0 0.017 0.0
-0.017 1.0 0.084
0.0 -0.084 1.0
</data>
</Matriz_Rotacion_Perfil>
<Contorno_Matriz_Perfil type_id="opencv-matrix">
<rows>556</rows>
<cols>2</cols>
<dt>i</dt>
<data> Aqui irían los puntos de las coordenadas que forman el contorno, que en este caso
serían 556 filas de puntos para la foto de perfil
</data>
</Contorno_Matriz_Perfil>
</opencv_storage>
```