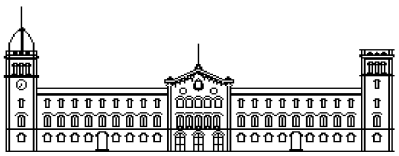


Treball fi de grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques
Universitat de Barcelona

Códigos correctores de cascadas de clasificadores y graph cuts para la segmentación multi-extremidad

---

Daniel Sánchez Abril

Directores: Sergio Escalera Guerrero
Miguel Ángel Bautista Martín
Realitzat a: Departament de Matemàtica Aplicada i Anàlisi. Universidad de Barcelona

Barcelona, 25 de juny de 2012

AGRADECIMIENTOS

Difícilmente hubiera conseguido realizar este proyecto sin ninguna ayuda y menos con un temario apenas visto en la carrera. Por esto, y solo esto, quiero demostrar mi gratitud a una serie de personas que me han respaldado en todo momento.

Primero a mis dos tutores: Sergio Escalera y Miguel Ángel Bautista. Ellos dos me han ido guiando a lo largo del extenso semestre con ideas para poder “atacar” el proyecto. Además han sabido motivar lo suficiente como para no perder las ganas de trabajar, aunque ha habido algún momento que otro que daba ganas de dejar las cosas como estaban y no volver durante un tiempecito.

Segundo, a mi compañero de proyecto Juan Carlos, que ha sabido explicarme más de una vez: ideas, conceptos, teoría del proyecto que me costaba entender por mi propia cuenta. Además, me ha salvado de varios apuros durante este proyecto.

Tercero, a mis compañeros de universidad por el apoyo moral y de trabajo para continuar trabajando lo suficiente como para tener buenas expectativas de cara al proyecto.

Cuarto… he tenido suerte de estar rodeado de esta gente.

"Twenty years from now you will be more disappointed by the things you didn't do than by the ones you did do. So throw off the bowlines, sail away from the safe harbor. Catch the trade winds in your sails. Explore. Dream. Discover." - Mark Twain

Resumen

El presente proyecto tiene como finalidad estudiar el funcionamiento y las posibilidades que ofrecen las diferentes técnicas del campo de la visión artificial para la detección y segmentación de extremidades de una persona sobre un gran volumen de imágenes.

La base de datos empleada contiene un gran número de imágenes dónde aparecen diversos sujetos realizando varias poses. De estos sujetos, se quiere detectar diferentes extremidades, tales como: cabeza, torso, antebrazos, brazos, muslos, o piernas.

Para poder realizar esta detección se ha optado por entrenar diferentes cascadas de clasificadores sobre características Haar sobre regiones orientadas. Una vez entrenadas, se les podrán enviar imágenes de la base de datos para detectar las extremidades ya mencionadas.

Además de utilizar las cascadas para la detección, se utilizarán unos métodos para corroborar si lo que detectan es cierto o no. Como siguiente paso se aplican técnicas de segmentación para alcanzar dos finalidades diferentes: segmentar únicamente el sujeto del fondo de la imagen y segmentar cada extremidad del sujeto con tal de separar el sujeto de la imagen pero con sus extremidades detectadas. En este caso se ha optado por segmentación mediante la formulación Graph Cuts. Los métodos presentados se han validado sobre un conjunto de datos realizados por un grupo de investigación del departamento, mostrando resultados satisfactorios.

Resum

El present projecte té com a finalitat estudiar el funcionament i les possibilitats que ofereixen les diferents tècniques del camp de la visió artificial per a la detecció i segmentació d'extremitats d'una persona sobre un gran volum d'imatges.

La base de dades emprada conté un gran nombre d'imatges on apareixen diversos subjectes realitzant diverses postures. D'aquests subjectes, es vol detectar diferents extremitats, com ara: cap, tors, avantbraços, braços, cuixes, o cames.

Per poder realitzar aquesta detecció s'ha optat per entrenar diferents cascades de classificadors sobre característiques Haar sobre regions orientades. Un cop entrenades, se'ls podran enviar imatges de la base de dades per detectar les extremitats ja esmentades.

A més d'utilitzar les cascades per a la detecció, s'utilitzaran uns mètodes per corroborar si el que detecten és cert o no. Com següent pas s'apliquen tècniques de segmentació per assolir dues finalitats diferents: segmentar únicament el subjecte del fons de la imatge i segmentar cada extremitat del subjecte per tal de separar el subjecte de la imatge però amb les seves extremitats detectades. En aquest cas s'ha optat per segmentació mitjançant la formulació Graph Cuts. Els mètodes presentats s'han validat sobre un conjunt de dades realitzats per un grup de recerca del departament, mostrant resultats satisfactoris.

Abstract

This project aims to study the performance and potential of different techniques in the field of artificial vision for the detection and segmentation of limbs of a person on a large volume of images.

The database used contains a large number of images where different subjects appear performing various poses. The main objective is to segment different limbs, such as head, torso, forearms, arms, thighs, or legs.

In order to perform this detection we train different classifiers cascades on Haar features on targeted regions. Once trained, they are used to detect the aforementioned extremities.

In addition of using the cascades for the detection, some methods will be used to verify if that detection is true or not. As a next step segmentation techniques are applied to achieve two different purposes: to segment only the subject from the background image and segment each limb of the subject provided to separate the subject of the picture but with his limbs detected. In this case we have chosen segmentation using Graph Cuts formulation. The methods presented have been validated on a dataset made by a research group of the department, showing satisfactory results.

ÍNDICE

RESUMEN DE FIGURAS

1. INTRODUCCIÓN

En este capítulo se realiza la introducción del proyecto, describiendo el contexto, motivación y el planteamiento de las necesidades detectadas que dieron origen a este proyecto. Además, se incluye un estudio del estado del arte sobre otras tecnologías/procesos/técnicas existentes en la literatura que abordan problemáticas similares.

1.1 Contexto

Una de las mejores maneras de poder detectar y analizar las extremidades de las personas es mediante la grabación de secuencias de vídeo en las que aparecen una serie de personas realizando movimientos, poses, etc. La clave radica en tener una amplia variedad de éstas para asegurar una alta generabilidad y discriminabilidad de los métodos de reconocimiento.

La eficacia en la detección de las poses es esencial ya que el análisis que se realizará a posteriori depende totalmente de éstas. Ya sea corriendo, saltando, agachado, riendo, chillando, etc. es válido cualquier tipo de pose que se pueda apreciar sin entrar en detalles.

En consecuencia, se ha obtenido de una base de datos de imágenes con un conjunto de sujetos realizando una serie de varias poses. Estos datos serán útiles para el análisis de diversas técnicas de visión por computador para reconocer las extremidades y las posiciones que ocupan según las poses. De esta manera, se podrán detectar las extremidades que nosotros queramos y estén en la imagen.

La metodología de detección de las extremidades abordada en este proyecto se basa en el aprendizaje por cascadas, que no es más que un conjunto de clasificadores encadenados. Para cada extremidad dispondremos de una cascada que aprenderá a reconocer dicha parte a partir de un gran conjunto de entrenamiento.

Además de tener las cascadas entrenadas, una forma de asegurarse de qué el resultado sea el esperado, es el uso de unos métodos de corrección de errores para poder corregir una posible desviación en el resultado.

Por otro lado, es también interesante no sólo detectar extremidades sino segmentarlas del fondo de la imagen. Para este propósito se utilizarán un conjunto de métodos de segmentación de visión por computador que permitirán segmentar la persona del fondo y distinguir el conjunto de extremidades del sujeto. De esta manera el resultado final del proceso no será sólo el sujeto separado del fondo de la imagen, sino que también se podrán diferenciar las extremidades de dicho sujeto a nivel de píxel.

1.2 Motivación

La motivación en este Proyecto Fin de Carrera (PFC) es por un lado, el diseño, implementación e integración modular de un sistema capaz de detectar y segmentar extremidades de personas dado un gran volumen de imágenes. Para ello se partirá de varios algoritmos representativos del estado del arte en esta cuestión que hemos considerado ventajoso en términos de eficiencia y calidad de los resultados.

La detección de gestos es un área de investigación desafiante que aborda el problema de detección de personas en las imágenes, la detección y descripción de las partes del cuerpo, dando a entender su configuración espacial, y la realización de la acción/detección de gestos en imágenes fijas o secuencias de imágenes. Debido a la enorme cantidad de poses que pueden realizar los humanos, recuperar las poses del cuerpo es un problema que tiene cierta dificultad e implica lidiar con varias distorsiones: los cambios de iluminación, las oclusiones parciales, cambios en el punto de vista de referencia, las deformaciones rígidas y elásticas, sólo por mencionar algunas. Incluso con la alta dificultad del problema, las técnicas modernas de visión por computador y las nuevas tendencias merecen más atención, ya que en los próximos años se esperan resultados prometedores. Por otra parte, varias sub-áreas han sido recientemente creadas, tales como: análisis de la conducta humana, la robótica social, etc. El esfuerzo que implica esta área de investigación se verá compensado por sus posibles aplicaciones: área de educación, investigación en el área de la sociología, vigilancia y seguridad, mejora de la calidad de vida a través del monitoreo o la asistencia artificial automática, etc. [1].

Además, diversas aplicaciones de este sistema pueden ser utilizadas en el ámbito de la medicina, por ejemplo: monitorizar las personas con discapacidad con el fin de detectar automáticamente los comportamientos de los pacientes en los hospitales, como el dolor o la agitación, extraer automáticamente un conjunto de indicadores de actitud y de comportamiento a fin de apoyar el diagnóstico de enfermedades mentales, etc.

En segundo lugar, intentaremos mejorar las técnicas, proponiendo una combinación idónea de las mismas para tener un sistema cuyos resultados sean los más satisfactorios posibles.

1.3 Estado del arte

La combinación de técnicas de visión por computador utilizadas en este proyecto no son las únicas y no necesariamente las óptimas, por este motivo vamos a comentar técnicas del estado del arte parecidas a cada una de las que hemos utilizado y así explicar los motivos que nos han llevado a usarlas. Conviene aclarar que las explicaciones dadas a continuación por cada técnica serán algo breves ya que más adelante se explicarán las elegidas con más detalle.

Las explicaciones que vienen a continuación vienen dadas por el orden en qué se implementan y conectan tales métodos. Primero de todo se extraen de la base de datos de imágenes sus características a partir del descriptor de características, para luego, enviar éstas a los clasificadores para que aprendan. Una vez los clasificadores están entrenados y se quieren probar, la forma de proceder será enviarles imágenes para validar si lo que detectan es correcto o no. Aquí es donde se utilizarán los códigos correctores de errores, para corregir los posibles errores a la hora de la clasificación. Por último, se utilizarán técnicas de segmentación para separar persona del fondo de la imagen y extremidades de dicho fondo.

1.3.1 Descriptor de características

Los descriptores visuales son herramientas muy necesarias para extraer la información que se considera importante de la imagen para que los clasificadores puedan realizar el aprendizaje. En este caso utilizaremos los descriptores HOG y Haar que son dos de las características más utilizadas en el estado del arte. Los HOG son histogramas de gradientes orientados que nos codifican información de las formas en imágenes. Son relativamente costosos de calcular y se suelen utilizar para aprender clasificadores discriminativos. En el segundo caso, los descriptores Haar se utilizan como derivadas discretas sobre imágenes y se usan para ser aprendidos por las cascadas mediante el clasificador Adaboost. Este punto se encuentra detallado en el trabajo del Sr. Juan Carlos Ortega en [17].

1.3.2 Clasificadores

Un clasificador es un sistema capaz de proporcionar una predicción de pertenencia a una clase como salida a partir de un conjunto de características tomadas como entradas. En la rama de aprendizaje supervisado, construir un clasificador se basa en definir una regla que pueda asignar una etiqueta a cualquier otro dato de entrada a partir de un conjunto de ejemplos etiquetados.

Un ejemplo de un clasificador es aquel que acepta datos de sueldos de una persona, edad, estado civil, dirección e historial de crédito y clasifica a la persona como aceptables o inaceptables para recibir una nueva tarjeta de crédito o préstamo.

Una de las posibles combinaciones de un conjunto de clasificadores es la cascada y ésta dará un mejor resultado dependiendo la cantidad de datos que se introduzcan, el número de clasificadores, etc.

La motivación para hacer uso de cascadas se debe a que éstas se usan para aprender conjuntos desbalanceados de datos. Por ejemplo, en el caso de querer aprender cabeza contra cualquier otro elemento del mundo visual, incluiremos en la cascada más imágenes de lo que no es cabeza. El uso de cascadas en este contexto nos permite hacer viable el aprendizaje desbalanceado a partir de la concatenación de clasificadores semi-balanceados. Esto se encuentra detallado en el proyecto del Sr. Juan Carlos Ortega [17].

1.3.3 Códigos correctores de errores

Para tratar problemas de multi-clasificación (donde el numero de clases $N>2$) de objetos o partes, la mayoría de técnicas usan votación. No obstante, los ECOC han demostrado que además de permitir votar entre múltiples clasificadores binarios también permiten corrección.

Los códigos de corrección de errores son un marco general para combinar clasificadores binarios a fin de abordar el problema multi-clase. En base a los principios de corrección de errores de teoría de la comunicación y debido a su capacidad para corregir los errores de la varianza de los clasificadores base, ECOC se ha aplicado con éxito a una amplia gama de aplicaciones de reconocimiento de patrones [2].

1.3.4 Graph cuts

Los métodos de segmentación que vamos a utilizar en las diferentes imágenes de la base de datos son dos: grabcut, que usa la teoría de saturación (algoritmo de Ford-Fulkerson) graph cut para la segmentación binaria, y alpha-expansion, que extiende la teoría de graph cuts al contexto multi-etiqueta (multi-extremidad en nuestro caso). El primero aborda la segmentación binaria, en nuestro caso lo utilizaremos para segmentar la persona del fondo de la imagen. Por otro lado, el segundo método es utilizado para realizar segmentación multi-clase, es decir, puede segmentar varias clases previamente especificadas. Éste lo utilizaremos para segmentar varias extremidades del cuerpo: brazos, piernas, cabeza, etc.

1.4 Propuesta

A partir de una base de datos de imágenes donde aparecen personas realizando varias poses, se desea aplicar técnicas de visión por computador para detectar las extremidades de dichas personas. Además, se quiere segmentar la persona de diferentes maneras para obtener dos resultados: separar la persona del fondo de la imagen y segmentar cada extremidad para tener el conjunto de extremidades de la persona separadas del fondo.

Todo este trabajo se realizará primeramente entrenando una serie de clasificadores en cascada, los cuales estarán asociados a unos descriptores de características que extraerán de cada imagen las características más importantes.

Una vez los clasificadores estén entrenados, los resultados que podamos obtener de ellos al enviarles imágenes de validación se les aplicará unos códigos para corregir los posibles errores a la hora de la clasificación.

Por último, se aplicarán dos métodos de segmentación para obtener los resultados ya comentados en el primer párrafo.

1.5 Objetivos

1. Usar descriptores visuales eficientes para extraer las características relevantes de las imágenes.

2. Entrenar diferentes clasificadores en cascada capaces de clasificar las extremidades de una persona.

3. Aplicar códigos de corrección de errores para corregir los posibles errores que los clasificadores puedan cometer.

4. Utilizar la segmentación binaria y multi-clase.

1.6 Estructura de la memoria

La memoria esta divida en las siguientes partes:

1. Descripción de la base de datos de imágenes utilizadas.
2. Explicación de los códigos correctores de errores.
3. Aclaración de la relación entre las cascadas y los códigos ECOC.
4. Explanación de la segmentación binaria.
5. Explicación de la segmentación multi-clase.
6. Resultados obtenidos.
7. Conclusiones.

2. BASE DE DATOS HuPBA

2.1 Introducción

En el campo de la Visión por Computador, uno de los problemas que ha adquirido más relevancia históricamente ha sido la detección de personas y el reconocimiento de la pose. Sin embargo, la literatura no aporta grandes bases de datos en este campo (tanto a nivel de tamaño como de protocolos de validación). Por eso, en este proyecto se propone la base de datos HuPBA desarrollada en la Universidad de Barcelona.

El mundo real donde nos movemos está lleno de objetos muy diferentes pero también de otros que son muy parecidos. El objetivo de este sistema es llegar a distinguir aquello que es una persona del resto de elementos visuales. Por lo tanto, tenemos que tener información (imágenes) de personas físicas como también de todo aquello que no es una persona, como pueden ser animales, objetos, etc.

Dentro del reconocimiento de las acciones humanas y su postura, hay una gran variedad de problemas a resolver como pueden ser describir sus partes, inferir su posición, reconocer los gestos que realizan, etc. A todo esto hay que sumarle las limitaciones del dispositivo con el que obtenemos la información del mundo, problemas en el punto de vista, problemas de iluminación, deformaciones… Por lo tanto, este es un campo donde existe una gran dificultad en el problema que se plantea y en el cual se investigan nuevas técnicas de Visión Artificial.

Los datos de las personas nos vienen dados por una base de datos desarrollada en la Universidad de Barcelona, llamada HuPBA (Human Pose and Behaviour Analisis). En nuestro caso, hemos seleccionado un subconjunto formado por 9 personas que han sido grabadas mediante una cámara digital, en forma de vídeo estando en distintas posturas y realizando distintos gestos. El proceso para tratar estos videos ha sido transformar éste en una sucesión de imágenes que posteriormente se tratarán para detectar sus partes o extremidades.

Posteriormente, para cada imagen (frame) de cada secuencia, se han etiquetado todas sus extremidades. Este proceso de etiquetaje se realiza mediante el uso de máscaras. Una máscara es una imagen en negro (ceros) de las mismas dimensiones que la imagen original pero que tienen una región blanca (unos) que indica donde se encuentra la extremidad que nos interesa. Por lo tanto, para cada secuencia, tenemos 14 directorios cada uno conteniendo máscaras para cada frame que indican donde se encuentran las extremidades del actor/actores en un determinado momento.

Estas regiones previamente definidas son las que utilizaremos para entrenar nuestros sistemas clasificadores, por lo que hace falta procesar toda la base de datos con sus máscaras e ir extrayendo todas las partes para posteriormente procesarlas.

La creación de esta base de datos fue realizada como proyecto final de carrera del Sr. Jordi Suñé Fontanals en 2011. El hecho de etiquetar tantas imágenes es una operación muy costosa, por lo que se intentó simplificar un poco mediante el uso de la interpolación de la posición de la extremidad. Esta técnica se basa en que se sabe que entre frame y frame la posición de las extremidades varía mínimamente. Teniendo en cuenta esta premisa, las extremidades se etiquetaron cada 2 frames, y mediante el uso de un programa en Matlab, se pudo completar la posición de las extremidades en los frames intermedios que no estaban

etiquetados. A continuación se muestra un ejemplo del contenido de la base de datos HuPBA:

Ilustración 1: Imagen original

Ilustración 2: Imagen de máscara asociada

Nuestra función ha sido procesar toda la base de datos, compuesta por 288827 imágenes y 4.5 Gb de capacidad aproximadamente. Hemos procesado y recortado la totalidad de las extremidades de todos los actores. Las partes del cuerpo que ha sido etiquetado son las siguientes:

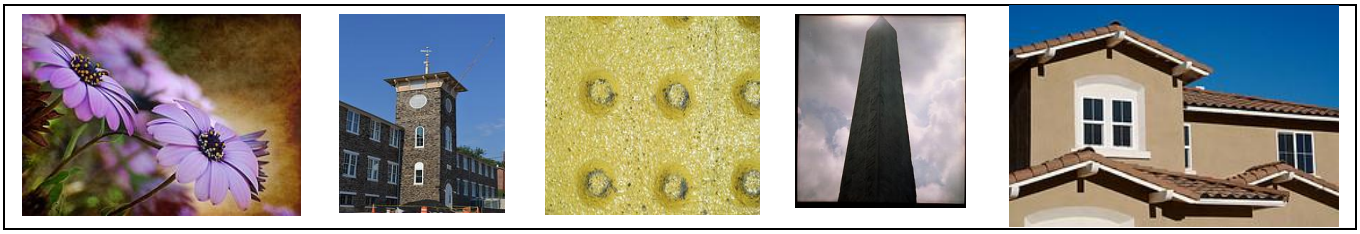
1. Cabeza	8. Brazo Derecho
2. Torso	9. Pie Izquierdo
3. Mano Izquierda	10. Pie Derecho
4. Mano Derecha	11. Pierna Izquierda
5. Antebrazo Izquierdo	12. Pierna Derecha
6. Antebrazo Derecho	13. Muslo Izquierdo
7. Brazo Izquierdo	14. Muslo Derecho

Una vez tenemos toda la información referente a la posición de las extremidades de las personas, nos hace falta incorporar a nuestro sistema todas aquellas imágenes que no son persona. El mundo real es un conjunto inmenso de elementos distintos, y que nuestros clasificadores tienen que aprender para distinguir las extremidades del fondo de una imagen, y este fondo puede ser cualquier imagen.

Para solucionar este problema, construimos una pequeña aplicación en Python, que se encargaba de hacer la tarea de crawler. Un crawler, o también llamada comúnmente “araña de internet” se encarga de inspeccionar un sitio web y de ir saltando a diferentes páginas mediante los hipervínculos que encuentra en la página actual, para seguir realizando un rastreo de la web de forma automática.

En nuestro caso, hemos programado un crawler que se encarga de obtener imágenes de la página web de Flickr. Mediante la creación de una cuenta de Flickr, podemos tener acceso a un sinfín de imágenes que abarcan todo tipo de temas, desde construcciones, coches inmuebles hasta animales, paisajes, objetos, etc. Todas estas imágenes nos servirán como conjunto de negativos y se consideran el fondo que los clasificadores tienen que aprender a distinguir de las extremidades.

Propusimos un filtro lo bastante general para obtener imágenes de todo tipo y a partir de aquí, fuimos descartando todas aquellas que contenían personas y niños, ya que esta información ya ha sido procesada anteriormente. En total, obtuvimos unas 2.500 imágenes finalmente filtradas de cualquier persona u extremidad. Este conjunto de imágenes los utilizaremos como conjunto de negativos, mientras que las imágenes de las extremidades serán el conjunto de positivos. Mediante diferentes técnicas que se explicarán más adelante, podremos conseguir que nuestro sistema aprenda a distinguir una extremidad del fondo de la imagen.
A continuación muestro algunas imágenes extraídas de Flickr y que han sido utilizadas para entrenar los clasificadores, para así detectar el fondo de la imagen:



Llegados a este punto, se nos presenta un problema bastante importante y que consiste en la orientación de las extremidades. Nuestra primera intuición nos diría que para obtener las extremidades simplemente tendríamos que realizar una operación AND entre la imagen original y la máscara, de tal forma que nos quedaría una imagen con la extremidad en cuestión. Pero esto no es del todo cierto, ya que las personas que hay en las imágenes están en distintas posturas y las posiciones de sus extremidades son diferentes. Esto es un problema, ya que pueden haber muchísimas combinaciones de orientación para una extremidad, lo que nos complica mucho más el proceso de aprendizaje, al no haber un patrón homogéneo.

Este problema ha sido solucionado mediante el análisis de un código en Matlab que obtiene la orientación predominante de una imagen. El funcionamiento consiste en analizar la región mediante las derivadas gausianas en X e Y. De esta manera se obtienen dos imágenes que nos indican el grado de cambio de intensidad dentro de la imagen. El paso siguiente es crear un histograma con todas las orientaciones obtenidas y quedarnos con la mayor. Este grado será el que nos indique cuántos grados tenemos que rotar la imagen.

Este proceso se realiza para tener todas las imágenes con una misma orientación, y de esta manera facilitar el proceso de entrenamiento de los clasificadores. Además, podemos evitar la confusión entre partes que se parecen mucho como son pies y las piernas.

El código estaba montado en Matlab, pero dado la gran cantidad de imágenes y extremidades que hay que procesar, al final decidimos pasar ese código a C/C++ mediante el uso de las librerías OpenCV y de algunas funciones ya implementadas. Después de algunas optimizaciones, el tiempo de ejecución por extremidad mejoró radicalmente.

Esta mejora de velocidad hizo que un proceso que tarda aproximadamente una semana en Matlab, se transforme en horas.

Lenguaje	**Tiempo de Ejecución por Extremidad**
Matlab	2.5 Segundos
C/C++	0.003 Segundos

2.2 Propósito de los datos

La parte de investigación de este PFC está enfocada en el análisis de los clasificadores en cascada (Cascadas AdaBoost) y SVM (Support Vector Machine). Un clasificador no es más que un sistema, el cual dará un resultado o predicción en función de su conocimiento y la información de entrada. Para realizar este proceso, hemos tenido que separar dos conjuntos de información:

1. Imágenes de Training (imágenes que se usarán en el entrenamiento del clasificador)
2. Imágenes de Test (Imágenes que se usarán para comprobar la validez del clasificador)

Por lo tanto, se ha realizado una organización de los datos. De todo el conjunto de imágenes de la base de datos, se han separado un conjunto para la fase de training de los clasificadores y otra para la parte de test de estos mismos. Por lo tanto, los datos de training no se pueden utilizar en la fase de test, ya que entonces nuestro sistema no serviría de nada. Siempre acertaría en la predicción ya que las imágenes de prueba forman ya parte del conocimiento de ese clasificador.

2.3 Procesamiento de los datos de entrada

Para cada actor, primero procesamos sus extremidades. En cada imagen o frame, seleccionamos la máscara asociada a esta y procedemos a obtener la información de la región que nos indica la máscara.

El proceso consiste en primero rotar y luego cortar, ya que si se realiza al revés, nos queda una imagen de la extremidad con bordes negros. Además, hay que sumarle a todo esto que las imágenes o regiones no son cuadradas. Por ejemplo, si tenemos una pierna etiquetada, su máscara tendrá una forma de rectángulo. Por lo tanto hay que obtener una región cuadrada de la misma longitud de lado, para tener todas las extremidades homogéneas.

El paso siguiente es analizar esa región con el algoritmo que calcula la orientación predominante y obtener el grado por el que hay que rotar la imagen. A continuación, tenemos que rotar la imagen de entrada, pero con el centro de rotación situado en el centro de la máscara. Finalmente realizamos el recorte de la imagen resultante, pero sin cambiar la posición de la nueva máscara cuadrada.

El resultado de este proceso nos devuelve todas las extremidades en imágenes cuadradas y que ya están rotadas. Estas imágenes serán las que utilizaremos para el primer paso del PFC, que será el entrenamiento de unas cascadas Adaboost con todas las imágenes de cada extremidad como positivos y las imágenes de fondo como negativos. El algoritmo es el siguiente:

1. Obtenemos la imagen actual original de la persona
2. Para cada extremidad, obtener su máscara correspondiente a la imagen original
3. Convertir la máscara a forma cuadrada
4. Realizar un primer recorte de la imagen correspondiente a la máscara
5. Calcular la orientación predominante de esa región
6. Con el grado obtenido anteriormente, rotar la imagen original como punto de rotación central que sea el centro de la máscara original cuadrada

7. Realizar un segundo recorte de la imagen original rotada y la máscara cuadrada sin rotar
8. Guardar la imagen de la extremidad recortada en su carpeta correspondiente

El hecho de optimizar todo este proceso en C/C++ fue una gran ventaja, ya que todas estas operaciones necesitaban de mucho tiempo en Matlab. El punto crítico lo tuvimos cuanto hicimos la conversión del código de obtención de la orientación predominante, ya que al estar montado en Matlab, no lo podíamos utilizar directamente. A continuación, en la Ilustración 3, muestro el funcionamiento por bloques de todo el sistema de una manera más detallada.

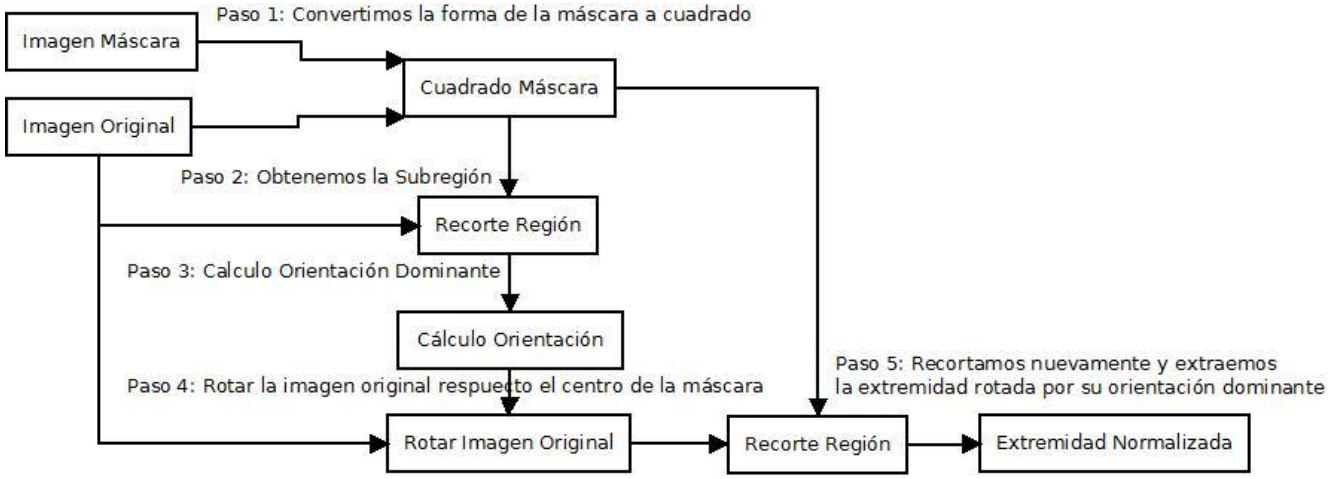


Ilustración 3: Proceso de extracción de extremidades

En nuestra implementación, hemos utilizado 6 extremidades. Hemos unido para cada extremidad sus componentes izquierda y derecha, en caso de tener para simplificar aún más y utilizar menos clasificadores. Ha habido componentes como las manos o los pies, que no hemos tenido en cuenta, ya que teníamos el problema de que el resultado recortado tenía unas dimensiones muy pequeñas. Esto obligaba a reescalar la imagen, con la perdida consiguiente de calidad. Finalmente, nos decidimos por incluir las 6 extremidades más importantes, y que son las siguientes:

1. Cabeza
2. Torso
3. Antebrazos
4. Brazos
5. Muslos
6. Piernas

El siguiente paso una vez tenemos todas las imágenes de las extremidades recortadas y procesadas es el entrenamiento de estas mediante las técnicas de Adaboost y SVM que a continuación explicaremos detalladamente.

3. MÉTODO

En este apartado se realiza una descripción detallada de cada método utilizado. Esto incluye tanto explicaciones teóricas, formulaciones como alguna implementación de código utilizada en el proyecto.

Primero se explicarán los códigos correctores de errores, también conocido como *ECOC* y sus pasos internos a la hora de utilizarlos. Una vez explicado estos códigos, se aclarará la relación que tienen éstos con las cascadas, exactamente la correspondencia entre la salida de la cascadas con la matriz *ECOC.* Una vez explicada la parte de la detección, quedarán por comentar los métodos de segmentación binaria y multi-clase aplicados a las imágenes de HuPBA.

3.1 ECOC: Error-Correcting Output Codes (Códigos de Corrección de Errores)

El análisis del marco ECOC ha demostrado que puede corregir los errores causados por los algoritmos de aprendizaje. Si un ejemplo es incorrectamente clasificado por alguno de los clasificadores aprendidos, todavía puede ser clasificado correctamente después de haber sido decodificado debido a la capacidad de corrección del algoritmo. ECOC es un marco de trabajo sencillo, pero potente, para hacer frente a los problemas de categorización multi-clase a partir de la combinación de clasificadores binarios.

Dado un conjunto de *N* clases, la base de los ECOC consiste en el diseño de un código por cada clase. Almacenando estos códigos en una matriz $M \in \{-1, 1\}^{N \times n}$ por filas siendo *N* el número de clases y *n* la longitud del código (numero de clasificadores). [3]. En la Fig.1 se puede ver un ejemplo de la estructura de una matriz ECOC. [4]

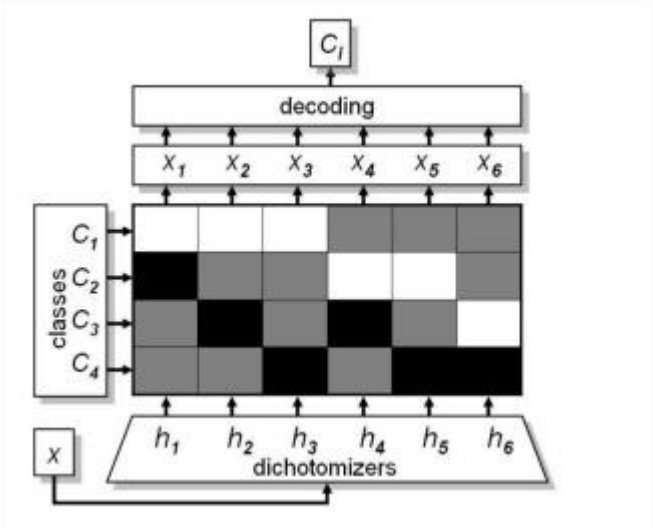


Figura 1. Ejemplo estructura matriz ECOC.

Desde el punto de vista de aprendizaje, *M* se construye considerando *n* problemas binarios, cada uno correspondiente a una columna de la matriz. Agregando clases en grupos (también conocidos como meta-clases), cada división define una partición de clases (codificados con +1 ó -1 dependiendo de su clase miembro, ó 0 si la clase no está considerada parte del problema binario). En la Fig. 2 se puede ver un problema binario entre tres clases con agrupaciones de uno-contra-uno [3].

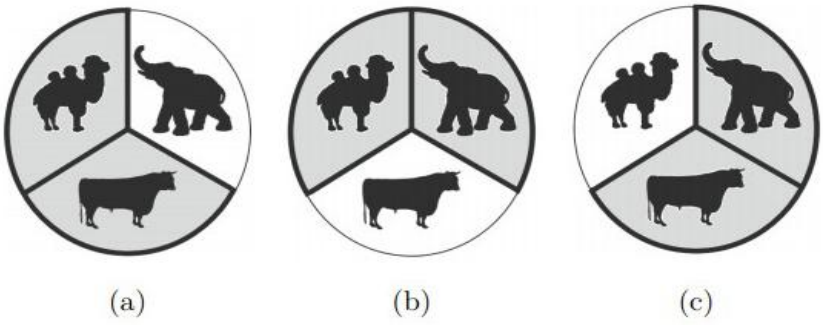
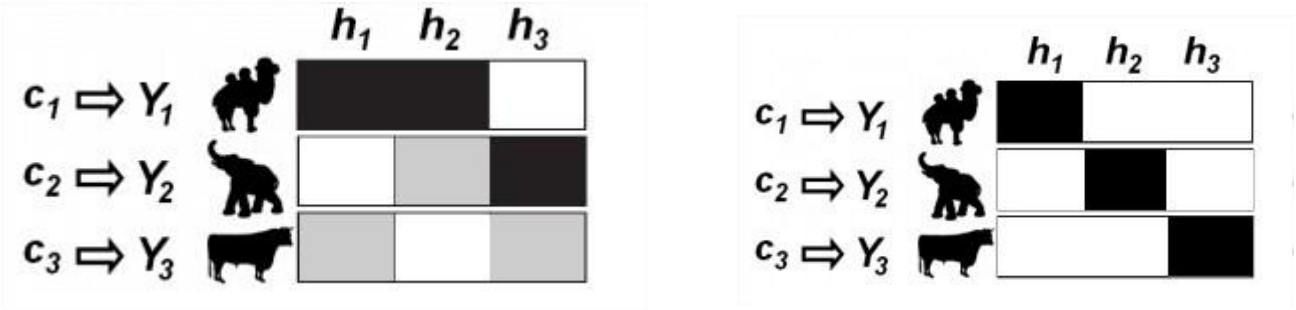


Figura 2. Problema binario entre 3 clases.

3.1.1 Codificaciones predefinidas

Las codificaciones estándar predefinidas están basadas en estrategias de uno-contra-uno o estrategias de uno-contra-todos. Una vez definidas las particiones de las clases, en la estrategia uno-contra-uno, cada una de ellas se codifica como una columna de la matriz *M* de codificación, como se muestra en la Fig. 3 (a). Las regiones negras son codificadas como 1 (división de las clases), y en blanco, regiones codificadas como -1 (segunda división de clases). Las regiones grises corresponden a las clases no consideradas por sus respectivos clasificadores. Ahora, las filas de la matriz *M* definen los elementos claves (Y_1, Y_2, Y_3) para sus clases correspondientes (c_1, c_2, c_3). En uno-contra-todos, cada clasificador (ó dicotomía) está entrenado para distinguir una clase del resto de clases. Dados *N* clases, está técnica tiene una longitud de código de *N* bits. Un ejemplo de la estrategia ECOC de uno-contra-todos para un problema de tres clases se muestra en la Fig. 3 (b) [3].



(a) (b)

Figura 3. Codificación ECOC. (a) uno-contra-uno. (b) uno-contra-todos.

3.2.2 Decodificación binaria

En el paso de la decodificación, aplicando los *n* clasificadores binarios entrenados, se obtiene un conjunto de predicciones para cada ejemplo del conjunto test. Este conjunto de predicciones se compara con la matriz *M* (fila a fila) utilizando una distancia δ, y se le asigna a la clase con la distancia mínima. El diseño de decodificación binaria más utilizada es la decodificación Euclidiana [3], por otro lado para la decodificación ternaria tenemos la decodificación Loss-Weighted.

1. *Decodificación Euclidiana:* esta métrica se basa en calcular la distancia Euclidiana de la palabra predicha por los clasificadores del ECOC con las diferentes filas de la matriz *M* [3]. En la Fig. 4 se puede ver un ejemplo de dicha decodificación donde la clase con distancia mínima es la del camello. Por tanto, la clase asignada será la del camello.

$$ED(x,y_i) = \sqrt{\sum_{j=1}^{n}(x^j - y_i^j)^2} \qquad [5]$$

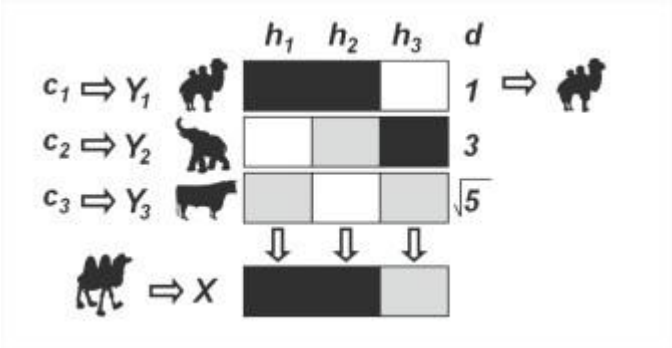


Figura 4. DECOC. Decodificación Euclidiana. [3].

2. *Decodificación Loss-Weighted*: esta métrica se basa en calcular una versión modificada de la distancia Euclidiana de la palabra predicha por los clasificadores del ECOC con las diferentes filas de la matriz *M.* De cara al proyecto, esta ha sido la decodificación seleccionada ya que los resultados que se pueden esperar son bastante altos.

$$d(\wp,i) = \sum_{j=1}^{n} M_W(i,j)L(M(i,j)\cdot f(\wp,j)) \qquad [6]$$

Cabe destacar, sin entrar en detalle, ya que esto se explicará con más claridad en el siguiente punto, que de las diferentes variables utilizadas en esta decodificación, hay una variable que representa unos valores obtenidos por las cascadas llamados pesos o *weights*. Estos pesos son útiles para optimizar los resultados durante la decodificación.

A continuación introducimos el algoritmo:
Dado un vector *p* obtenido por las predicciones de las cascadas, se multiplicará cada elemento de *p* por cada elemento del vector fila *Mij, donde M* es la matriz ECOC y cada fila es el vector clave y_i. A esto, también hay que multiplicarle el peso correspondiente a la cascada. Una vez hecho todos los productos, estos se sumarán para tener como resultado una distancia *d.*

Estos pasos se tiene que realizar para cada fila de la matriz *M,* dando como resultado tantas distancias *d* como filas y_i. Para acabar, se seleccionará la clase relacionada al vector fila *yi* que tenga la distancia más pequeña.

3.2 Caso cascada – ECOC

Una vez explicada la matriz ECOC, su codificación y decodificación, a continuación hablaremos sobre el caso particular de dichas fases en nuestro proyecto. Además, comentaremos las cascadas utilizadas, ya que su salida se conecta con la entrada de la matriz ECOC.

3.2.1 Cascada

Los clasificadores en cascada utilizados son un *boosting* de 8 niveles, es decir, cada cascada utiliza 8 clasificadores y cada uno usa como descriptor de características el método *Haar.* El número total de cascadas utilizadas son 6 (una para cada extremidad): cabeza, cuerpo, antebrazos, brazos, muslos y piernas. Cada cascada aprende dicha extremidad del resto, además de añadir el fondo.

3.2.2 ECOC – codificación

Tal como hemos avanzado un poco en los puntos anteriores, la codificación utilizada ha sido uno-contra-todos (*one-versus-all*). En este apartado detallaremos como se ha implementado dicha codificación.

La mejor forma de diseñar el ECOC ha sido conceptualmente pensar en que cada extremidad tiene que diferenciarse de las demás, o lo que es lo mismo, clasificar cada extremidad teniendo en cuenta que las demás extremidades y el fondo no pueden confundirse con la extremidad a clasificar. Por ejemplo, en el caso de la cabeza, queremos que la cabeza se diferencie, sin ninguna ambigüedad, del cuerpo, brazos, fondo, etc. ya que confundir la cabeza con los muslos sería algo ilógico y significaría que el diseño pensado no es lo suficientemente robusto.

Teniendo en cuenta los anteriores requisitos procederemos al análisis de las distintas codificaciones para seleccionar aquella que se ajuste más a nuestras pretensiones.

El diseño de ECOC que mejor se ajusta, como hemos comentado anteriormente, es el llamado uno-contra-todos. Este método ofrece la posibilidad de plasmar en ECOC nuestra premisa de diferenciar una extremidad de todas las demás.

	Cabeza	Cuerpo	Antebrazos	Brazos	Muslos	Piernas
Cabeza	1	-1	-1	-1	-1	-1
Cuerpo	-1	1	-1	-1	-1	-1
Antebrazos	-1	-1	1	-1	-1	-1
Brazos	-1	-1	-1	1	-1	-1
Muslos	-1	-1	-1	-1	1	-1
Piernas	-1	-1	-1	-1	-1	1
Fondo	-1	-1	-1	-1	-1	-1

Figura 5. Matriz ECOC utilizada.

La Fig. 5 muestra la matriz ECOC implementada donde las filas de la primera columna representan las clases utilizadas, y las columnas de la primera fila contienen el nombre de las cascadas (o *dicotomías*) usadas.

Como podemos comprobar, el nombre de unos-contra-todos queda reflejado al ver las columnas de la matriz ECOC, donde únicamente encontramos un 1 (lo que se quiere clasificar) y todo lo demás son -1 (lo que no se quiere clasificar).

3.2.3 ECOC - decodificación

Tal como hemos avanzado un poco en los puntos anteriores, la decodificación utilizada ha sido mediante *Loss-Weighted.* En este apartado detallaremos la salida de las cascadas, sus pesos y como se ha implementado la decodificación ECOC.

Salida de las cascadas:

Cada cascada puede enviar dos tipos de resultados:

1: Significa que ha detectado su respectiva extremidad.
-1: Significa que no ha detectado su respectiva extremidad.

Una vez se sabe la salida de todas las cascadas, esta formará un vector *X* de longitud 6 que se enviará a la decodificación ECOC para calcular la correcta salida o lo que es lo mismo, a que correcta extremidad pertenece o a qué extremidad se acerca/parece más.

Pesos de las cascadas:

Para la decodificación, se han utilizado unos pesos (o *weights*) generados por cada cascada. Estos pesos son un componente importante para este tipo de decodificación, ya que sin ellos el resultado variaría bastante y perdería parte de su eficacia. Los pesos están calculados a partir de una serie de valores que cada cascada va generando mientras realiza su aprendizaje para luego utilizar estos valores en la siguiente fórmula y obtener los pesos:

$$\frac{\prod_{i=1}^{N} P_i + \prod_{i=1}^{N} (1 - N_i)}{2}$$

La Fig. 6 muestra los pesos calculados para cada cascada:

Cabeza	Cuerpo	Antebrazos	Brazos	Muslos	Piernas
0.534205	0.404527	0.245545	0.232889	0.274047	0.307478

Figura 6. Pesos de las cascadas.

Una vez comentada la salida y pesos de las cascadas, a continuación se explican los pasos para la decodificación basada en *Loss-Weighted*.

Pasos decodificación ECOC:

1. Primero de todo, para cada fila M_i de la matriz ECOC, se multiplicará cada componente M_{ij} con el componente del vector X_j y por un peso asignado a cada cascada P_j, para luego restar los 6 resultados (ya que la fila es de longitud 6).

2. A continuación, se suman en valor absoluto cada componente de la fila M_i, es decir, se suman todos los -1's y el 1 de la correspondiente fila.

3. Una vez tenemos estos dos componentes, se realiza una división del primer resultado entre el segundo. Este resultado representará la distancia, o mejor dicho, "lo cerca que está el vector *X* de esa extremidad".

4. Una vez se ha realizado este cálculo para cada fila de la matriz ECOC, se escogerá la distancia más pequeña de entre las 7(7 filas de ECOC) ya calculadas. La distancia más pequeña significa que es la que se parece más/está más cerca de *X*.

5. En caso de qué la distancia más pequeña se repita, que es posible, el desempate a realizar será escoger de entre esas distancias pequeñas, una de forma aleatoria.

6. Por acabar, se devolverá el vector M_i respecto la distancia mínima.

3.3 GrabCut

En este punto se detallan las bases del grabCut (GMM y graph cut). Además, se añadirán los pasos seguidos en su implementación y los resultados obtenidos.

3.3.1 GMM: Gaussian Mixture Model (Modelos de Mezclas de Gaussianas)

De toda la base teórica que tiene grabCut, uno de los métodos más importantes es el *Gaussian Mixture Model,* también conocido como *GMM.* Éste consiste en una función de densidad de probabilidad paramétrica representada como la suma ponderada de densidades de componentes Gaussianas [7]. Estas componentes se combinan para proporcionar una densidad multimodal y pueden ser empleadas para modelar los colores de un objeto con el fin de realizar tareas, tales como: seguimiento (o *tracking)* de color en tiempo real y segmentación. Estas tareas pueden hacerse más robustas mediante la generación de un modelo de mezclas correspondiente a los colores del fondo (*background)*, además de un modelo para los colores del primer plano (*foreground)*, y emplear el *teorema de Bayes* para realizar clasificación por píxel [8].

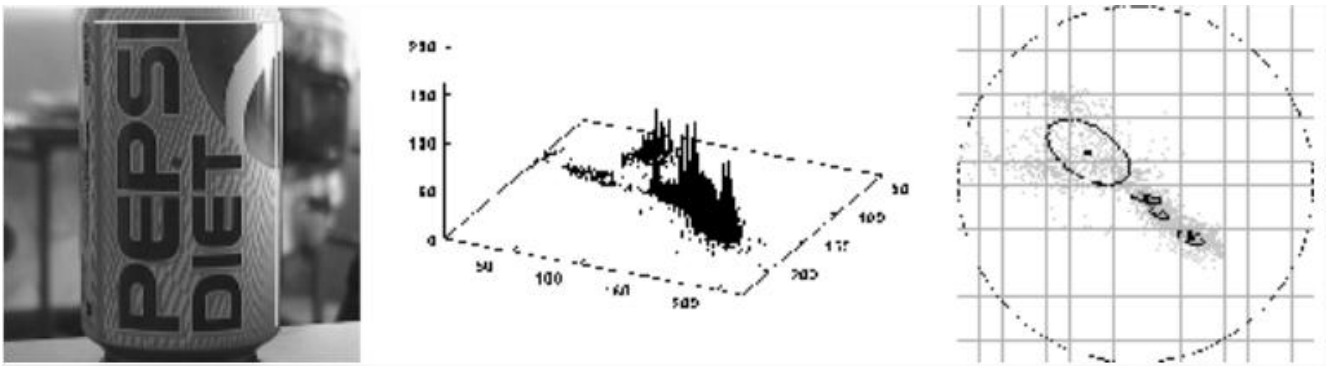


Figura 7. Representación *GMM.* (a) Objeto multicolor (una lata de Pepsi). (b) Histograma de color del objeto. (c) GMM del objeto. [8].

La Fig.1 muestra un objeto multicolor del cual se genera su histograma y *GMM* correspondiente. Cabe señalar que su histograma de color sólo es viable cuando una gran cantidad de datos están disponibles. Respecto a su *GMM,* los componentes de la mezcla se muestran como contornos elípticos de igual probabilidad.

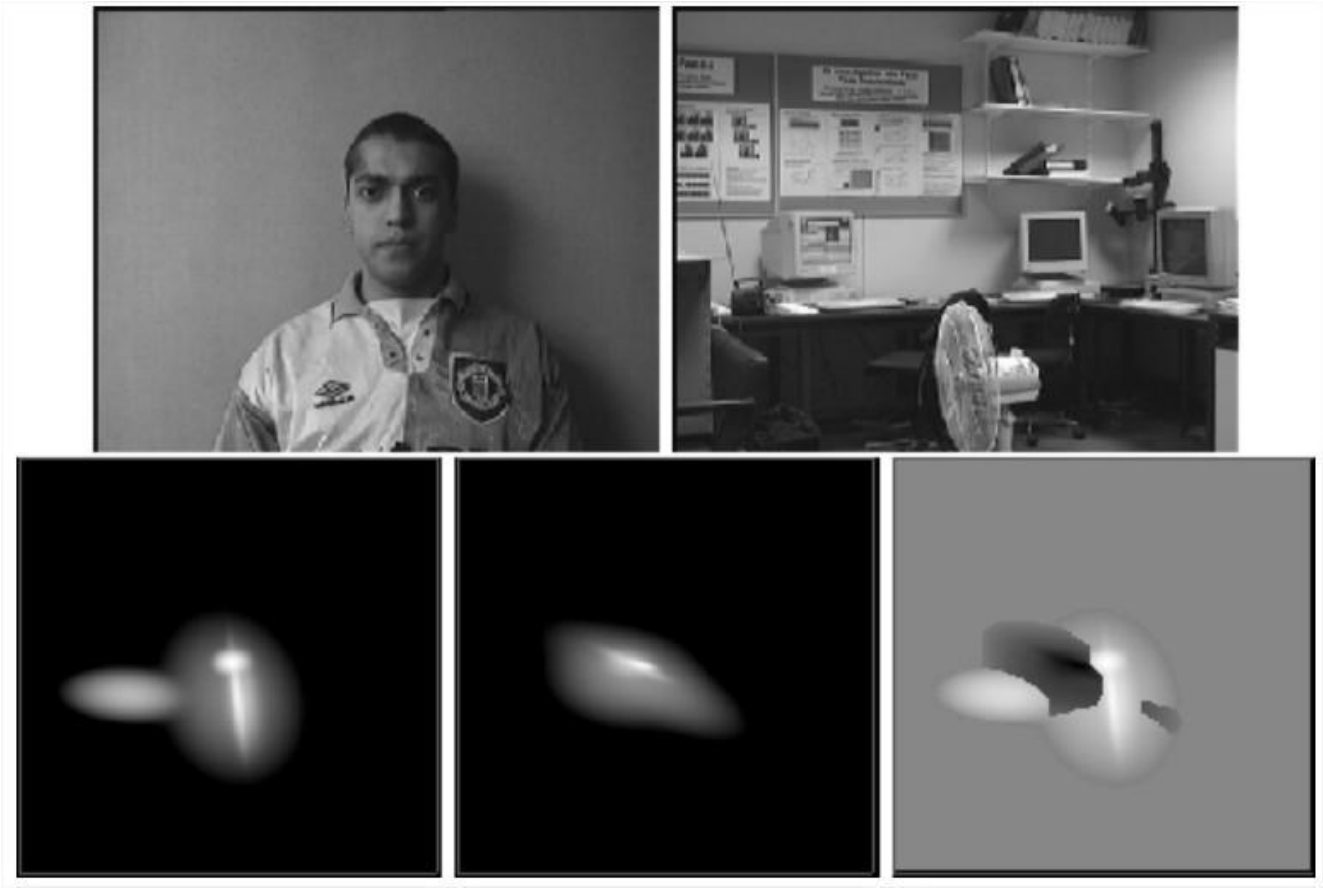


Figura 8. Modelos de mezcla de color de un objeto multicolor (modelo de la persona) y el contexto (modelo de la escena). (a) *Foreground*. (b) *Background.* (c) Densidad de probabilidad del *foreground.* (d) Densidad de probabilidad del *background*. (e) Combinación de las dos densidades. [8].

La Fig.8 (a) y (b) muestran las imágenes utilizadas para construir los modelos *foreground* (persona) y *background* (laboratorio). Por otro lado, en la parte (c) y (d) se puede observar la densidad de probabilidad estimada para los modelos de mezcla *foreground* y *background*, respectivamente. Por último, la parte (e) ilustra la combinación de las dos densidades donde las regiones más brillantes hacen referencia al *foreground* mientras que las regiones oscuras representan el *background.* Las áreas grises son regiones que no se han podido determinar.

A menudo, *GMM* se utiliza para *clustering*. Los *clusters* son asignados al seleccionar el componente que maximiza la probabilidad posterior. Al igual que *k-means*, *GMM* utiliza un algoritmo iterativo que converge a un óptimo local. Además, *GMM* puede ser más apropiado que el *k-means* cuando los *clusters* tienen diferentes tamaños y correlación dentro de ellos [9].

3.3.2 Graph cuts

El método en que se basa *grabCut,* es el llamado *graph cut* (o corte de grafos) que se implementó por primera vez en el campo de visión por computador por Greig, Porteus i Seheult de la Universidad de Durham, aplicándolo sobre imágenes binarias sobre las que se pretendía suavizar el ruido de imagen de mala calidad. Al tratarse de imágenes binarias, la aplicación del algoritmo produjo una solución exacta [10].

Graph cut consiste en un método de segmentación de imágenes basado en regiones que puede ser utilizado para resolver de manera eficiente una amplia variedad de problemas de bajo nivel en la visión por computador, tales como: suavizar imágenes, el problema de correspondencia estereoscópica, asignación de etiquetas (tales como: intensidad, disparidad, regiones segmentadas) a píxeles, etc. [11].

A continuación se explican los pasos del método *graph cut:*

1. *Segmentación de imágenes:* División de una imagen en regiones conexas i disjuntas relacionadas con los objetos de la escena i que cumplan un cierto criterio de homogeneidad.

2. *Clasificación de regiones:* Identificar cada una de las regiones de una partición como parte de una clase o categoría (segmentación con regiones no necesariamente conexas).

3. *Etiquetado de regiones:* Asignar una etiqueta (un color) diferente a cada componente conexa.

4. *Conectividad entre píxeles*: Dos píxeles de un conjunto están conectados si existe un camino que une los dos píxeles, de tal forma que todos los píxeles del camino estén en el mismo conjunto.

    a. *Conectividad 4* (la utilizada en segmentación): Cada píxel tiene 4 vecinos, en vertical i horizontal.
    b. *Conectividad 8*: Cada píxel tiene 8 vecinos, en vertical, horizontal i diagonal.

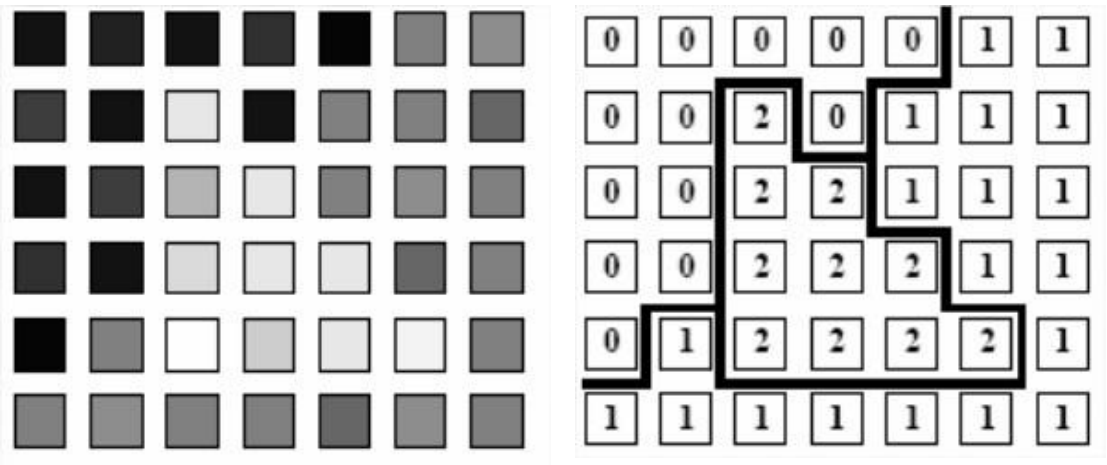


Figura 9. Etiquetado de una imagen. (a) Imagen en escala de grises. (b) Asignación etiqueta. [11].

La Fig. 9 ilustra la asignación de etiquetas a los píxeles de una imagen en escala de grises dependiendo su intensidad. Si nos fijamos en la imagen, podemos observar, en general, que predominan tres intensidades: negro, gris y gris claro. A estos tres tipos de intensidades se les asigna una etiqueta para diferenciarlas de las demás. Una vez todos los píxeles están asignados a sus respectivas etiquetas, ya disponemos de tres tipos de clases diferentes.

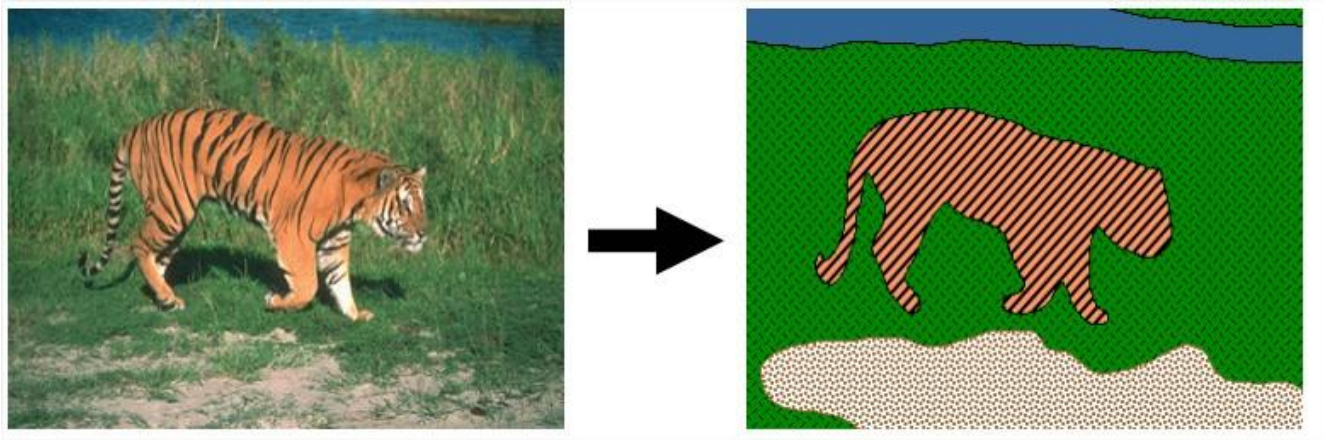


Figura 10. Ejemplo al usar *graph cuts.* (a) Imagen original. (b) Imagen resultante.

Uno de los posibles resultados que se pueden obtener al aplicar *graph cuts* es el de la Fig. 10, la cual muestra las posibles segmentaciones halladas entre el tigre y el entorno.

3.3.3 Descripción

Este método ha sido desarrollado por Carsten Rother, Vladimir Kolmogorov y Andrew Blake en el año 2004 [12].

El método *grabCut* está categorizado dentro de la segmentación binaria, es decir, cuando únicamente hay que separar dos clases. La forma de separarlas es mediante la extracción eficiente e interactiva de un objeto del primer plano en un entorno complejo, cuyo fondo no puede ser eliminado de forma trivial. Con lo cual, tendremos el objeto a recortar o segmentar, y por otro lado, el fondo de la imagen, cuyos resultados quedarán reflejados, en que esas dos clases se representan como dos tipos de etiquetas diferentes devueltas por *grabCut*.

Dado que la implementación que se dio en su día fue muy bien aceptada por la simpleza en qué el usuario únicamente tenía que seleccionar el objeto a recortar y el fondo, el objetivo se convirtió en lograr un alto rendimiento con un esfuerzo modesto por parte del usuario.

Según se puede observar en la Fig. 11, se compara el método respecto a otros, donde la interacción del usuario utilizando *GrabCut* es únicamente para marcar un rectángulo conteniendo la zona de interés, mientras que en el resto, el usuario ha de esforzarse más para marcar puntos o zonas del objeto en cuestión [13].

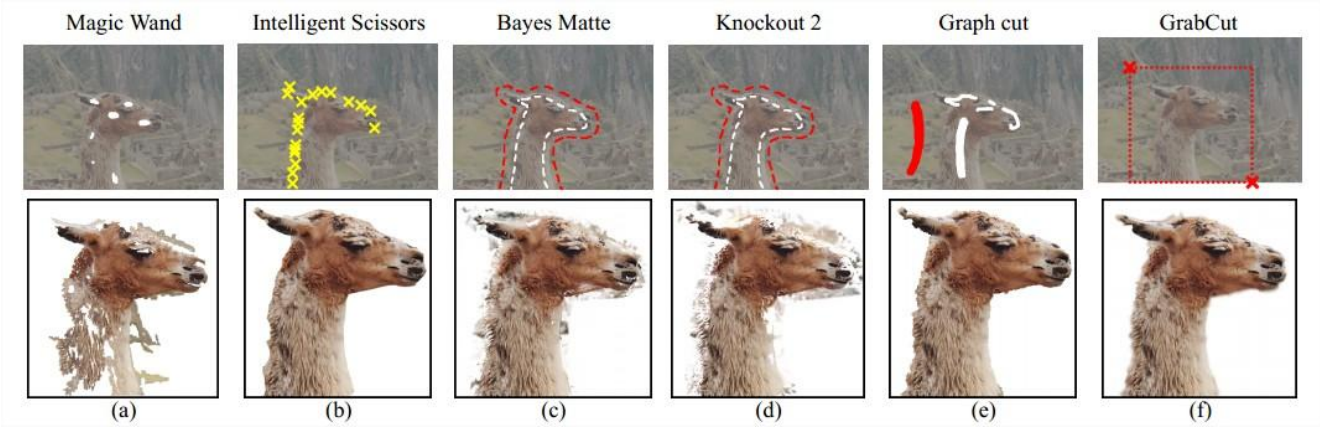


Figura 11. Comparación de métodos de segmentación [12].

Además, la segmentación realizada por los métodos *Intelligent Scissors, Graph cut* y *GrabCut* es bastante óptima, pero es importante reseñar que en caso de *grabCut* el usuario casi no necesita interactuar.

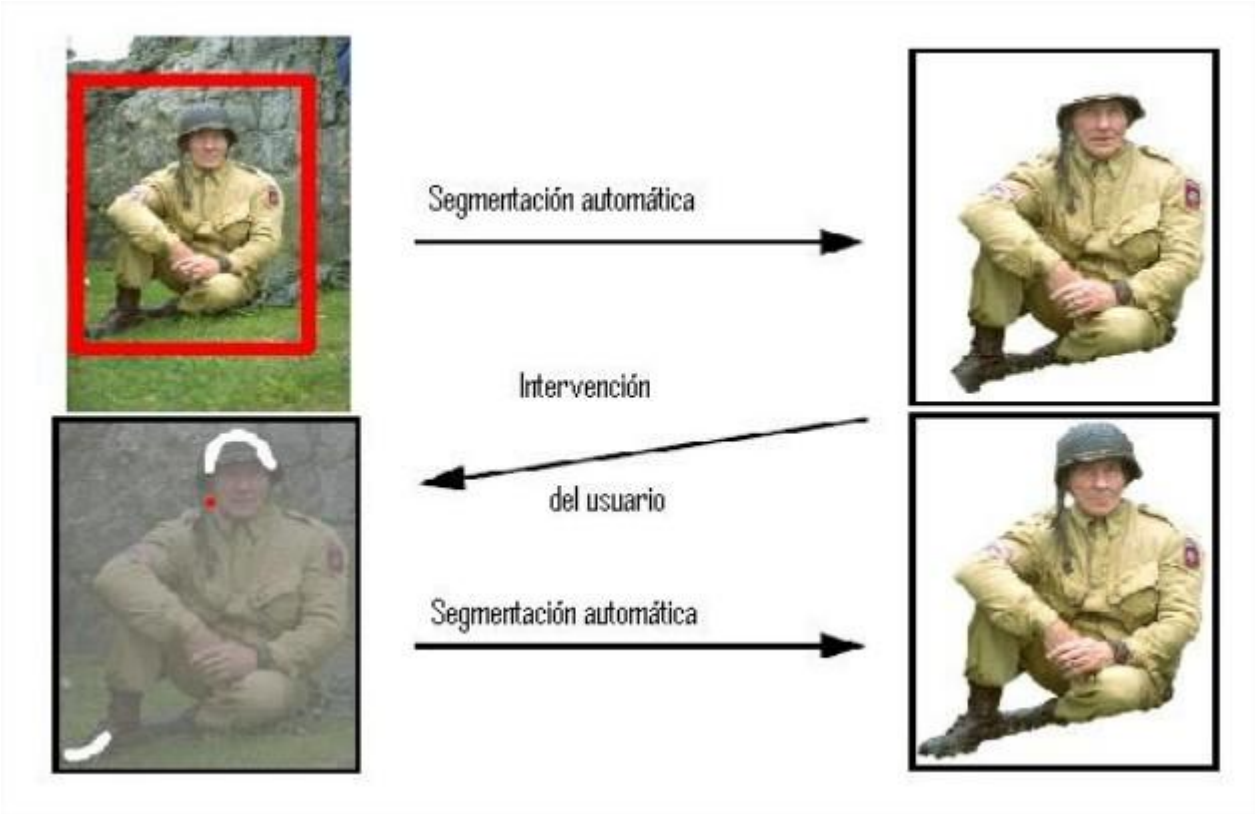


Figura 12. Proceso de intervención del usuario. [14].

Adicionalmente, si el usuario desea, puede seleccionar áreas de la imagen manualmente para mejorar el resultado inicial, como se observa en la Fig. 12. Si observamos la Fig. 12 de izquierda a derecha y de arriba abajo, encontramos que la primera imagen es el resultado de seleccionar con el ratón el recuadro del objeto a segmentar, mientras que la segunda imagen es el resultado de la segmentación. Después, el usuario puede seleccionar áreas manualmente para mejorar la segmentación, tal como muestra la tercera imagen y su resultado en la cuarta imagen.

3.3.4 Implementación

En el comienzo de este proyecto, se ha utilizado GrabCut para evaluar su calidad, eficiencia y qué resultados permite generar. Visualmente los resultados son aceptables, con una segmentación muy ajustada a los bordes de la imagen. La implementación de este algoritmo viene codificada como ejemplo de las librerías OpenCV [15].

Dado que junto a las librerías se facilita todo el código fuente que las forman, incluyendo los ejemplos de las diferentes funcionalidades, se modificó para permitir que fuese posible su uso sin la interacción del usuario, haciendo posible que se llame al programa mediante la imagen a segmentar y otra imagen (máscara) que indica qué zonas pertenecen a cada objeto. Así resulta menos costoso, más fácil y exacto evaluar el proceso, ya que no hace falta elegir todo el rato las partes que pertenecen o no al objeto.

Figura 13. Resultado obtenido mediante método *grabCut.* (a) Imagen a segmentar. (b) Imagen segmentada.

Un pequeño ejemplo se puede ver en la Fig. 13, la cual muestra la segmentación de la persona con unos bordes bastante ajustados y casi sin pérdida de la información. Lo único que aún quedaría por segmentar sería el fondo que hay entre las piernas de la persona pero es algo que el algoritmo utilizado no es capaz de solventar.

Para comprender mejor como se realiza el análisis, se puede visualizar el siguiente diagrama en la Fig. 14, donde se ve reflejado el bucle de funcionamiento.

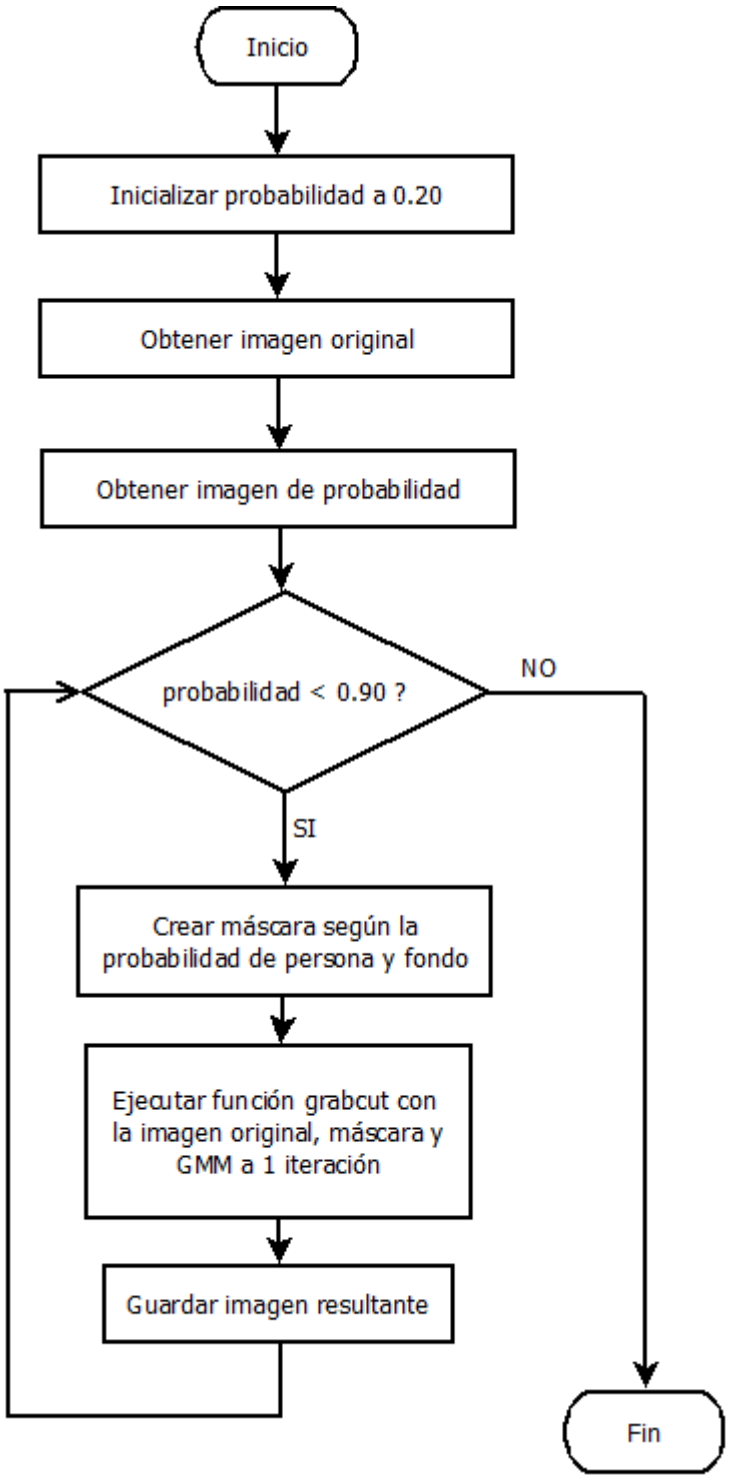


Figura 14. Diagrama de flujo para las imágenes aplicando *grabCut*.

3.4 Multi-label alpha-beta-swap graph cuts

En este punto se explica el funcionamiento de *graph cuts multi-label*, es decir, aplicar la segmentación a una imagen cuando hay más de dos clases en juego, además del algoritmo que logra realizar tal segmentación, llamado *alpha-beta-swap.* Este algoritmo se encarga de optimizar y segmentar la imagen realizando un etiquetado de los diferentes píxeles que la forman. Este etiquetado se realiza permitiendo que un gran número de píxeles cambien sus etiquetas simultáneamente.

Una vez se haya explicado la base teórica que hay detrás, se detallará la implementación llevada a cabo para el método *multi-label* acompañado de algún resultado como ejemplo.

3.4.1 Graph cuts multi-label

En el caso de la multi-segmentación, se busca poder dividir la imagen original en un número de partes mayor a 2. Uno de los objetivos de este proyecto es la identificación de las diferentes extremidades de una persona, tales como: cabeza, torso, brazos etc.

3.4.2 Alpha-beta-swap

Este algoritmo se caracteriza por mover algunos píxeles etiquetados α a la etiqueta β pero a su vez, algunos píxeles etiquetados como β se etiquetan como α. Es decir, se intercambian (Swap) píxeles entre dos etiquetas.

En este caso tenemos dos etiquetas o disparidades α y β asociadas a dos píxeles distintos. El método consiste en intercambiar los pixeles etiquetados como α por los β y viceversa de manera que disminuya la función de energía.

El algoritmo se resume en los siguientes pasos, ver Fig. 15:

1. Comenzar con una etiqueta arbitraria f
2. Establecer éxito= 0
3. Para cada par de etiquetas α y β
   a) Encontrar $f'' = argminE(f')$ siendo f' el valor de la etiqueta al producir un único intercambio entre α y β
   b) Si $E(f'') \leq E(f)$ entonces establecer $f = f''$ y éxito= 1
4. Si éxito= 1 volver al paso 2
5. Devolver el resultado final de la etiqueta f

Figura 15. Algoritmo alpha-beta-swap.

En la figura siguiente Fig.16, se puede visualizar un ejemplo de este algoritmo. La primera imagen muestra el etiquetado inicial, donde hay 3 clases. La segunda imagen visualiza el proceso de intercambiar una etiqueta en un paso. La tercera enseña cómo se cambian un gran número de etiquetas simultáneamente.

Figura 16. Imagen resultado de aplicar alpha-beta-swap. (a) Etiquetado inicial. (b) Movimiento estándar. (c) Aplicación alpha-beta-swap [10]

3.4.3 Implementación

Para poder implementar este algoritmo, se ha utilizado la librería gco-3.0 realizada por Olga Veksler y Andrew Delong [16].

Esta librería se basa en una parte codificada en forma de librerías en ANSI-C/C++, que una vez compilada generan unos archivos ejecutables utilizables desde Matlab. La otra parte es un “wrapper”. Son una serie de archivos que se encargan de recibir las llamadas realizadas en código Matlab y traspasarlas a los archivos ejecutables.

En la siguiente página se encuentra la Fig. 17, que muestra el diagrama de flujo de la aplicación para generar las imágenes resultantes de aplicar el algoritmo alpha-beta-swap.

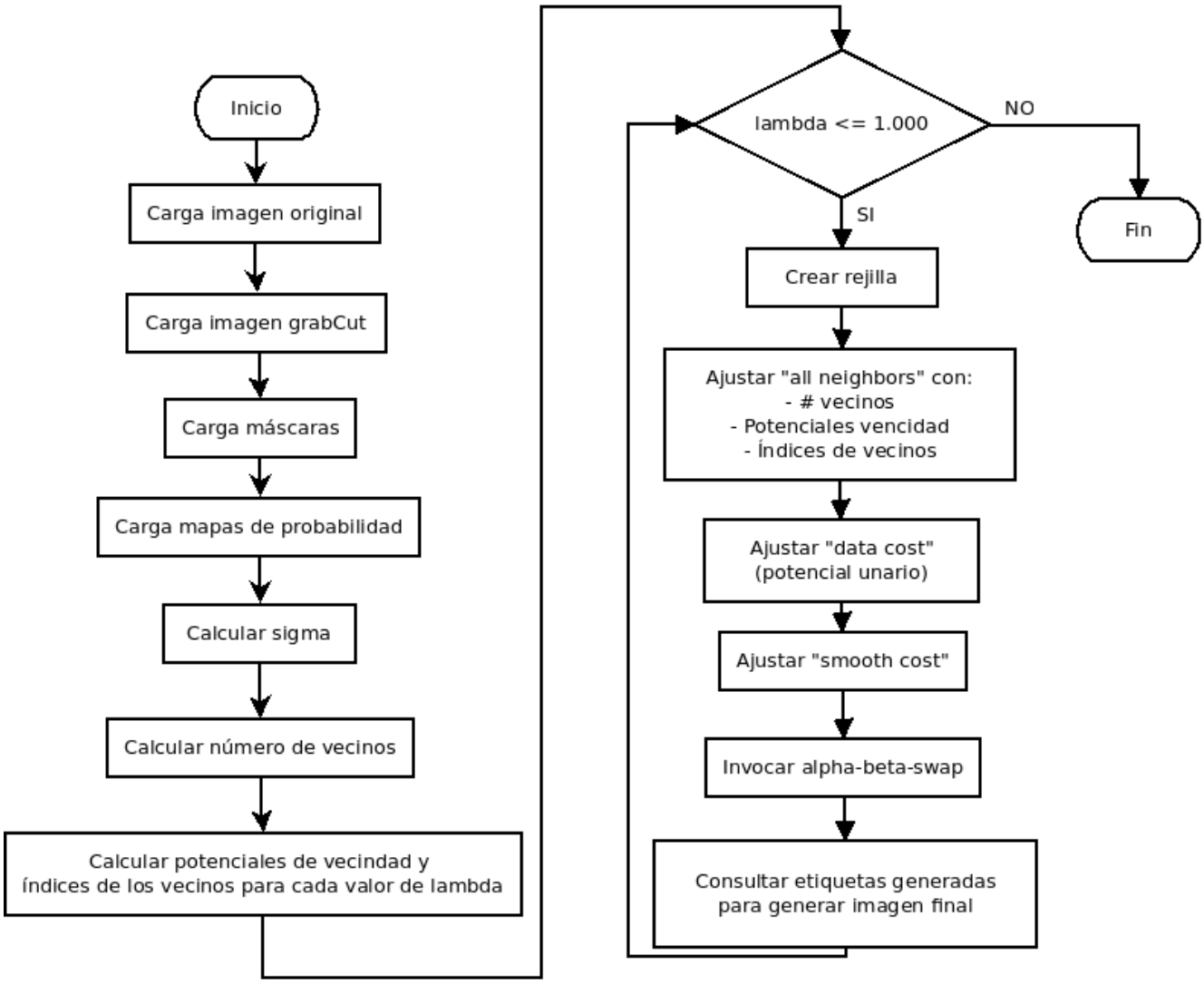


Figura 17. Diagrama de flujo para las imágenes aplicando alpha-beta-swap.

La explicación del funcionamiento es la siguiente:

1. El primer paso es cargar la imagen original en escala de grises.
2. A continuación hay que cargar la imagen del grab cut para generar su respectiva máscara del ground-truth y realizar la operación AND de ésta y la imagen resultante del multi-label. De esta manera se obtendrá el ground-truth de la persona segmentada con todas sus posibles extremidades.
3. Cargar todas las máscaras de la base de datos correspondiente a la imagen original. Este paso se realiza para calcular el acierto del algoritmo sobre la imagen original.
4. Otras imágenes importantes son los mapas de probabilidad obtenidos al aplicar el sliding window sobre la imagen original e ir obteniendo el tipo de detección para cada región y realizar una votación a posteriori. Esta votación hace que se obtenga los valores del mapa de probabilidad para cada extremidad.
5. Este paso consiste en calcular un valor llamado sigma, el cual forma parte de la fórmula para calcular los potenciales de vecindad. El cálculo consiste en sumar las distancias entre los vecinos del píxel en cuestión con dicho píxel y teniendo en cuenta que la relación vecino-píxel no esté repetida. Una vez se realice la suma de todos los píxeles, se dividirá entre el par de vecinos no repetidos. Es decir, si se ha contabilizado el par de vecinos (p1, p2), no habrá que contar (p2, p1) ya que en temas de distancias es lo mismo y la relación es recíproca.

6. Calcular el número de vecinos para cada píxel.
7. Se calcula los potenciales de vecindad y los índices de los vecinos. La siguiente fórmula muestra cómo calcular dichos potenciales:

$$\lambda * \exp(-(Wpixel - Wneighbor)^2/\sigma)$$

donde λ: [1, 10, 100, 1000]

Para cada valor de λ:

8. A partir del número de etiquetas y píxeles, crear la rejilla con estructura 8 vecinos máximo por píxel.
9. Ajusta todos los vecinos a partir de: número de vecinos, potenciales de vecindad e índices de vecinos.
10. Calcula el data cost cumpliendo las siguientes condiciones:
    - Para todas las etiquetas que no sean fondo y el píxel detectado en la imagen grabcut sea fondo, se asigna un coste de 1000. En caso contrario, es decir, el píxel detectado en la imagen grabcut sea persona(pertenezca a alguna de las extremidades), se asigna un coste mediante la siguiente fórmula:

      Decir que los valores de los píxeles de los mapas de probabilidad son potenciales en vez de coste y están en un rango [0, 255]. Uno de los requisitos para que se puedan utilizar satisfactoriamente en la fórmula es que estén en un rango [0, 1], por este motivo se normaliza mediante la división entre el valor máximo del mapa de probabilidad de la extremidad en cuestión. El paso de potencial a coste se realiza mediante la función *log().*

    - En caso de que la etiqueta sea fondo y el píxel detectado en la imagen grabcut sea fondo, se asigna un coste de 0. En caso contrario, es decir, el píxel detectado en grabcut sea persona, se asigna un coste de 1000.
11. Ajustar smooth cost.
12. Llamar a *alpha-beta-swap*.
13. Se consulta a la rejilla por sus diferentes etiquetas para a continuación, dibujar en una imagen de salida la representación de dicha etiqueta con un color que la diferencie de las demás.
14. Como resultado, se habrán generado 4 imágenes (1 para cada valor de λ).

En la Fig. 18, se puede ver como ejemplo uno de los diferentes resultados obtenidos en este proyecto. En este caso se puede apreciar que detecta relativamente bien la cabeza y una parte de las piernas. En cambio, las demás extremidades no las ha podido detectar bien.

Figura 18. Imagen resultante al aplicar alpha-beta-swap. (a) Imagen original. (b) Imagen resultado.

4. RESULTADOS

Una vez definidos los objetivos de este proyecto y explicado los diversos métodos utilizados, además de su implementación, se procede a detallar los resultados obtenidos.

Los resultados se explicarán siguiendo el *pipeline* del proyecto: *grabCut* y *multi-label*. Del primer método se mostrarán diversas imágenes óptimas y algunas no tanto para comparar. Por último, en el apartado *multi-label* se ilustrará las imágenes con las diferentes etiquetas obtenidas y una métrica para comprobar el porcentaje de aciertos del algoritmo.

4.1 GrabCut

A continuación se detallan los parámetros utilizados y los resultados obtenidos.

4.1.2 Parámetros

Para la implementación del método *grabCut* se han tenido que configurar ciertos valores para que los resultados fueran lo suficientemente aceptables. Igualmente, también se detallarán algunos parámetros que dan como resultado imágenes poco aceptables para después comparar.

A continuación se explican los parámetros utilizados:

1. Imagen original de la base de datos.

Figura 19. Imagen original.

2. Imagen de probabilidad.

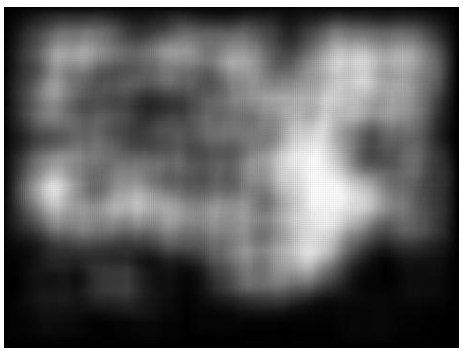


Figura 20. Imagen de probabilidad.

La imagen de probabilidad es resultado de varios pasos que se detallan a continuación:

1. Creación de una matriz con el mismo tamaño que el de la imagen original. Esta matriz representará el mapa de probabilidad.

2. Para cada región obtenida del *sliding window*, las cascadas detectan a qué extremidad se acerca más y el resultado obtenido es clasificado entre persona o fondo.
3. En caso que sea persona, se realizará una votación que consiste en sumar "+1" a todos los píxeles de la imagen que contengan dicha región. De esta manera obtendremos como su nombre bien indica un mapa de probabilidad en el que cada pixel tendrá una probabilidad de ser o no ser persona.

4. En caso que sea fono, no se realiza dicha votación.

5. Normaliza la imagen para que los valores estén en el rango [0, 1].

Dado que las librerías *OpenCV* no permiten guardar imágenes con un rango de valores [0, 1], éstas se tienen que pasar a *UINT8* para luego ser guardaras a disco.
Dichas imágenes son las que se muestran en este proyecto.

3. Antes de explicar los parámetros de este punto, se debe aclarar a que se debe su uso, ya que si no, no tiene sentido para el lector.

   La función *grabCut* implementada utiliza una matriz de probabilidades parecida a la que se utiliza en este proyecto. La única diferencia es que esta matriz tiene que fijarse con unas constantes exigidas por la función *grabCut.*

   Estas constantes son:

   - *GC_BGD*: Se sabe con seguridad total que es fondo.
   - *GC_PR_BGD*: Es probable que sea fondo.
   - *GC_PR_FGD*. Es probable que sea persona.

   Para los diferentes valores del mapa de probabilidad creado en los puntos anteriores, se deben establecer los rangos a los que pertenecerán las constantes explicadas arriba. Por ejemplo, si hay valores entre 0 y 0.20 se puede asignar la constante *GC_BGD*. Esto significará, de manera entendible, que las probabilidades que estén en dicho rango (el cuál es realmente bajo) serán con total seguridad fondo.

   Esos rangos son los que se han de parametrizar y son los que pertenecen al punto 3.
   - GC_BGD para [0, 0.10]
   - GC_PR_BGD para [0.10, PROB]
   - GC_PR_FGD para [PROB, 0.9]

   La constante *PROB* es inicializada a 0.20 con un valor de incremento de 0.1 hasta llegar a 0.9. Esto se ha realizado de esta manera para generar todas las imágenes *grabCut* posibles y quedarnos con la que segmenta mejor. Para unos valores de PROB bajos, el algoritmo no suele segmentar bien. En general, la segmentación óptima empieza cuando *PROB* vale 0.5. Más adelante se mostrarán unos ejemplos para luego explicar en detalle los motivos del mal resultado.

4. El número de iteraciones fijadas en la función *grabCut* ha sido de 1. Se ha comprobado a lo largo de diversas pruebas, que el algoritmo a partir de una iteración ya segmenta de manera óptima. Para iteraciones mayores a 5, 10… los resultados son prácticamente idénticos al de 1 iteración.

4.1.2 Resultados cualitativos

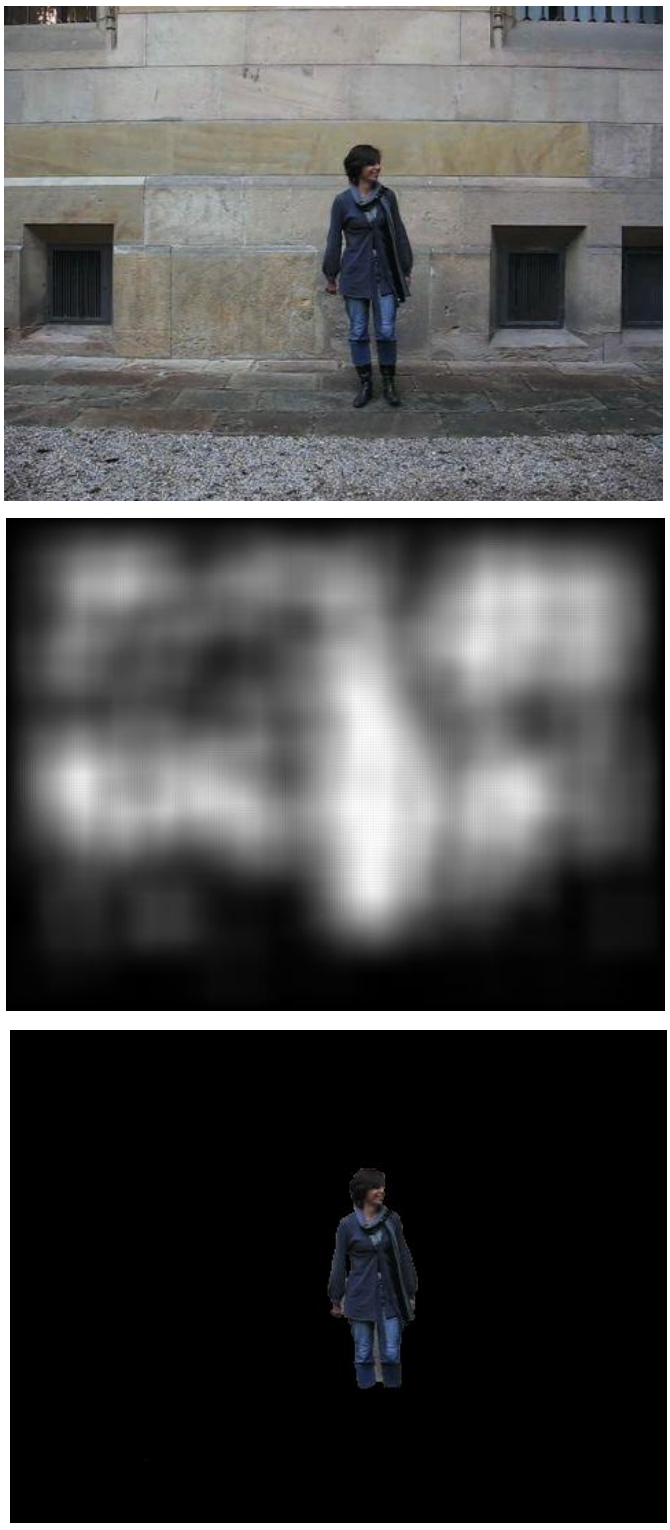


Figura 21. Resultado *grabCut* 1

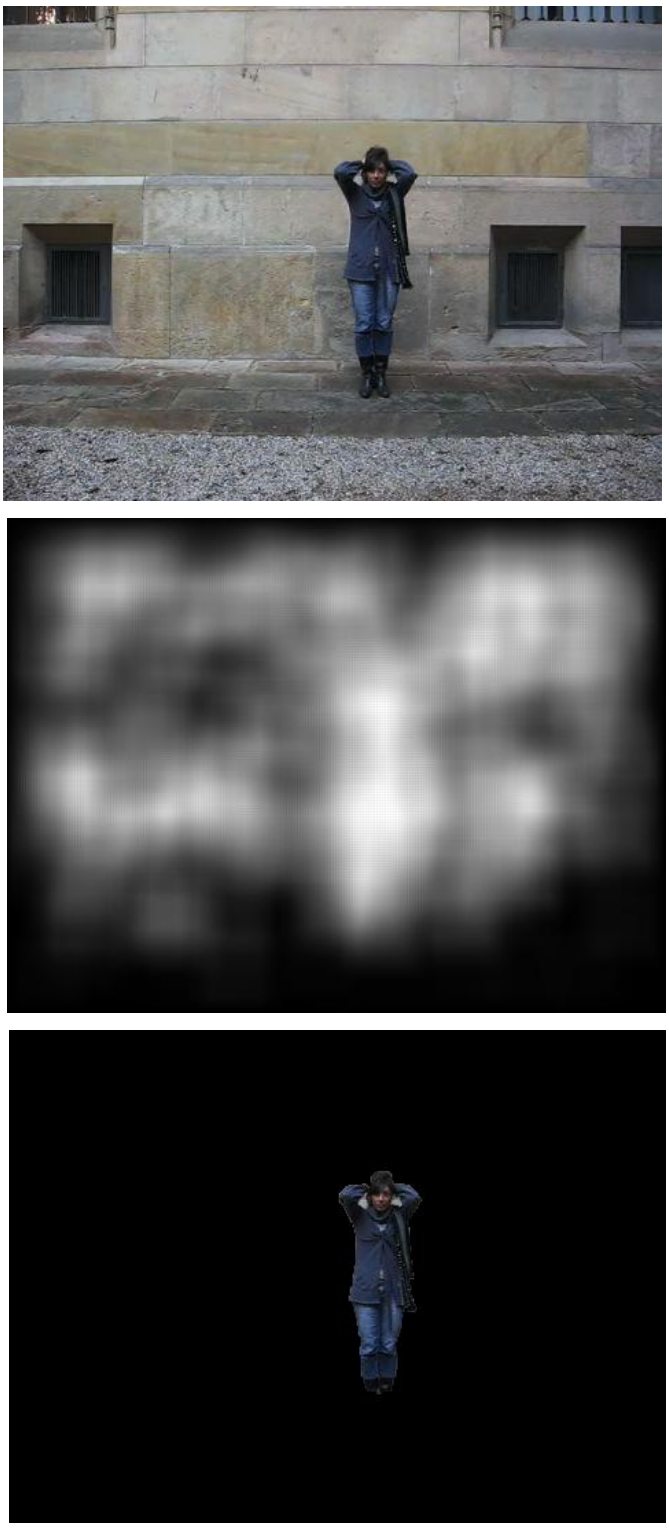
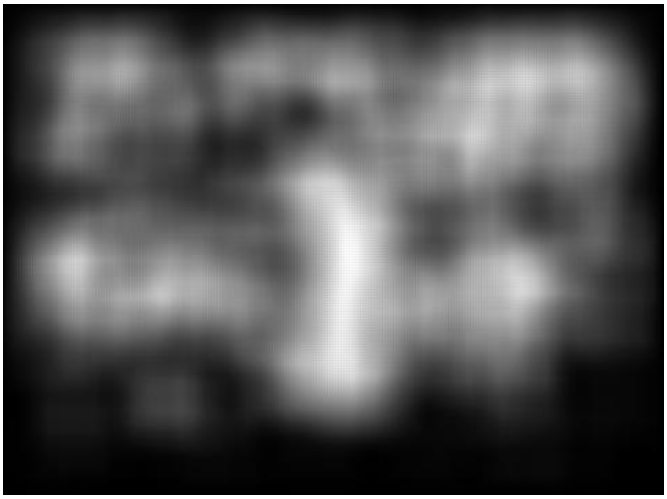
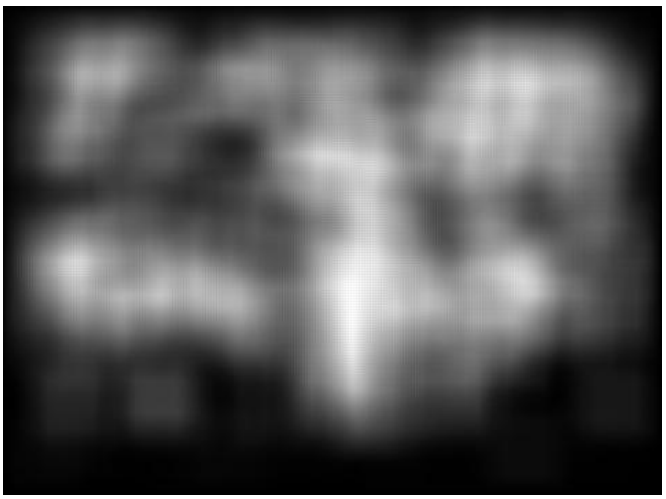


Figura 22. Resultado *grabCut* 2





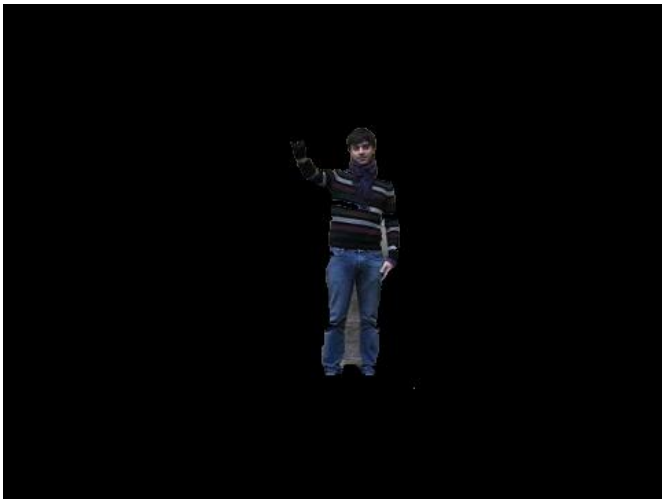
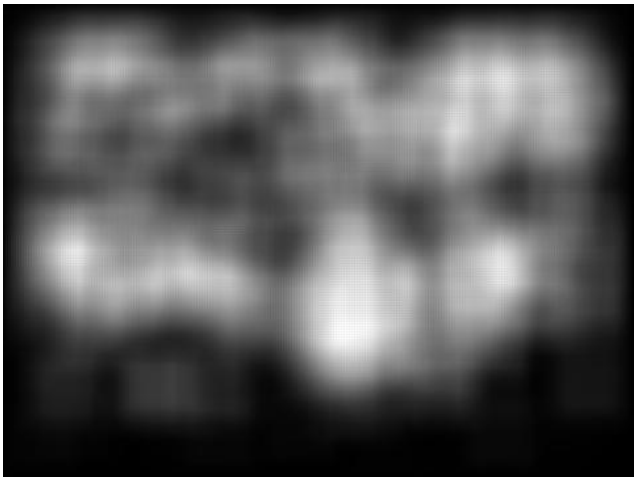
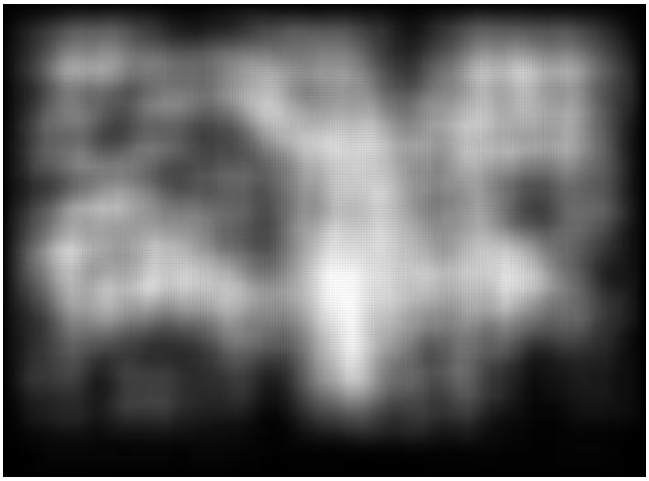


Figura 23. Resultado *grabCut* 3

Figura 24. Resultado *grabCut* 4





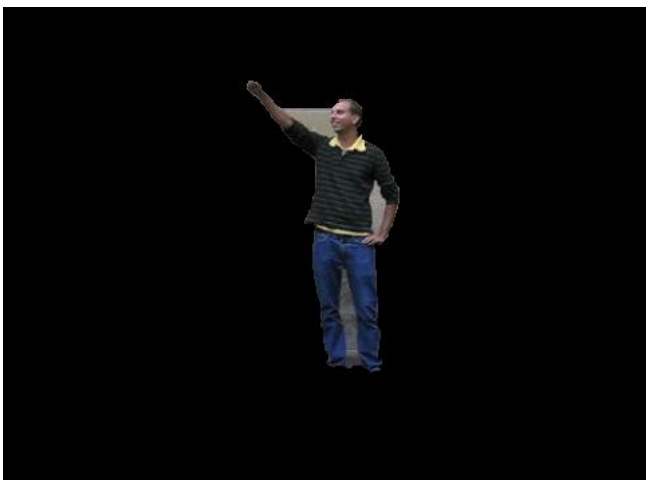


Figura 25. Resultado *grabCut* 5

Figura 26. Resultado *grabCut* 6

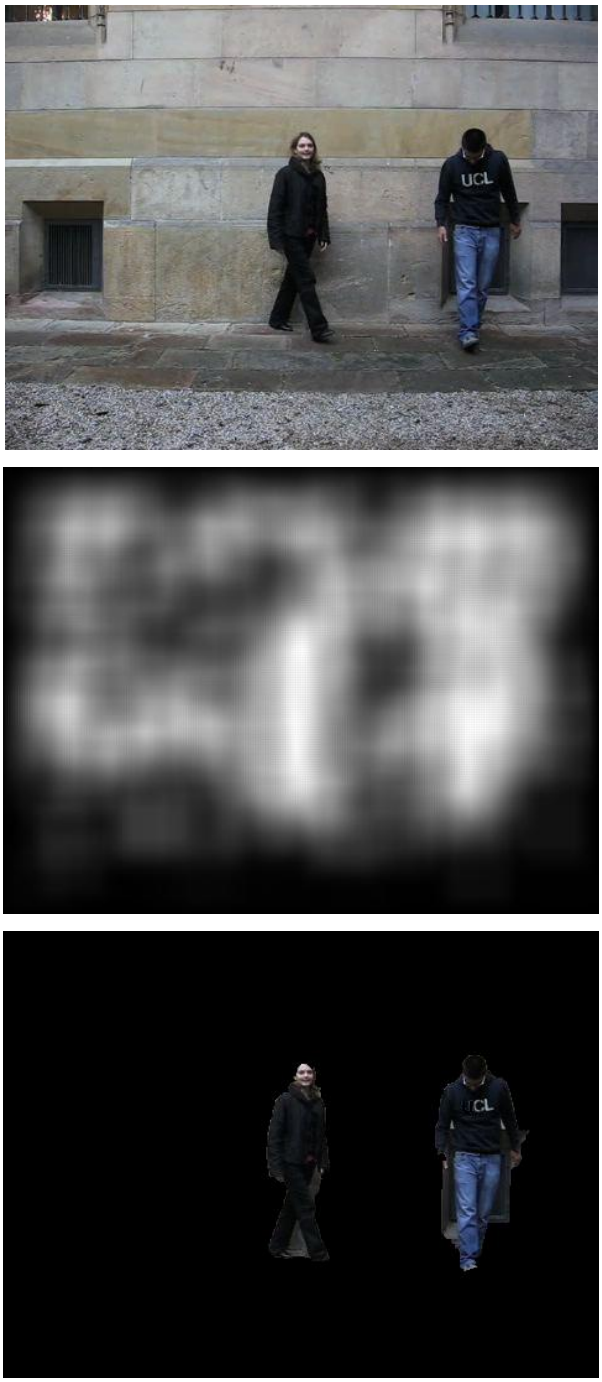


Figura 27. Resultado *grabCut* 7

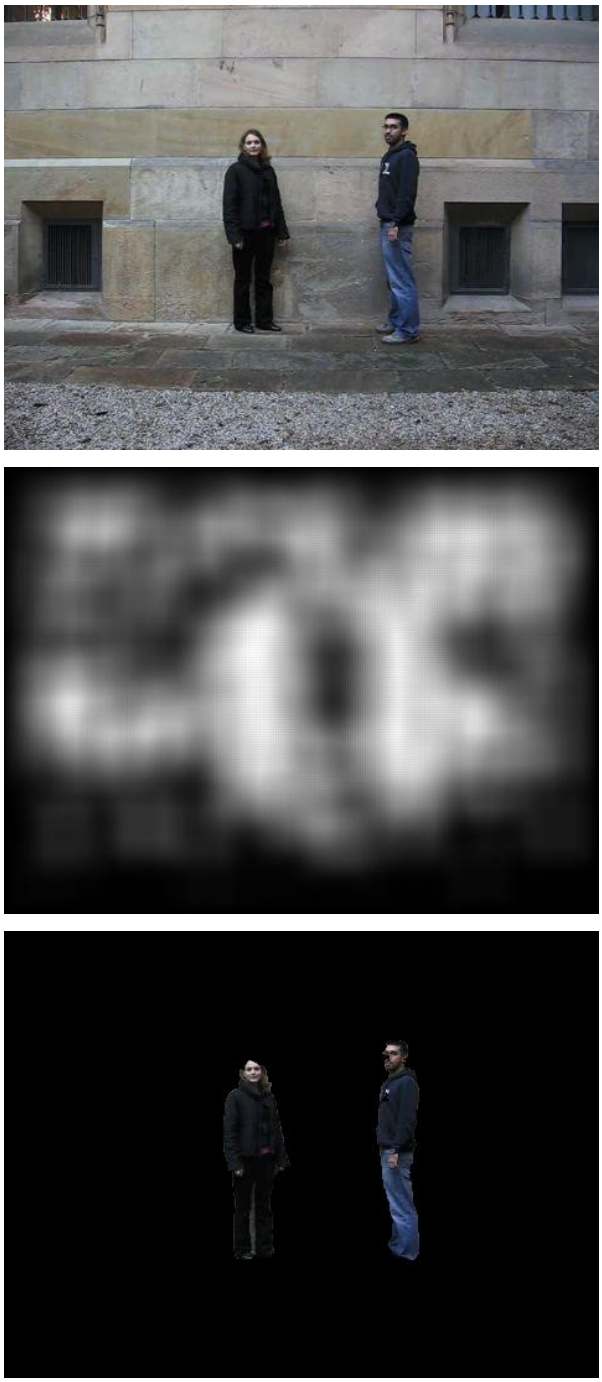


Figura 28. Resultado *grabCut* 8

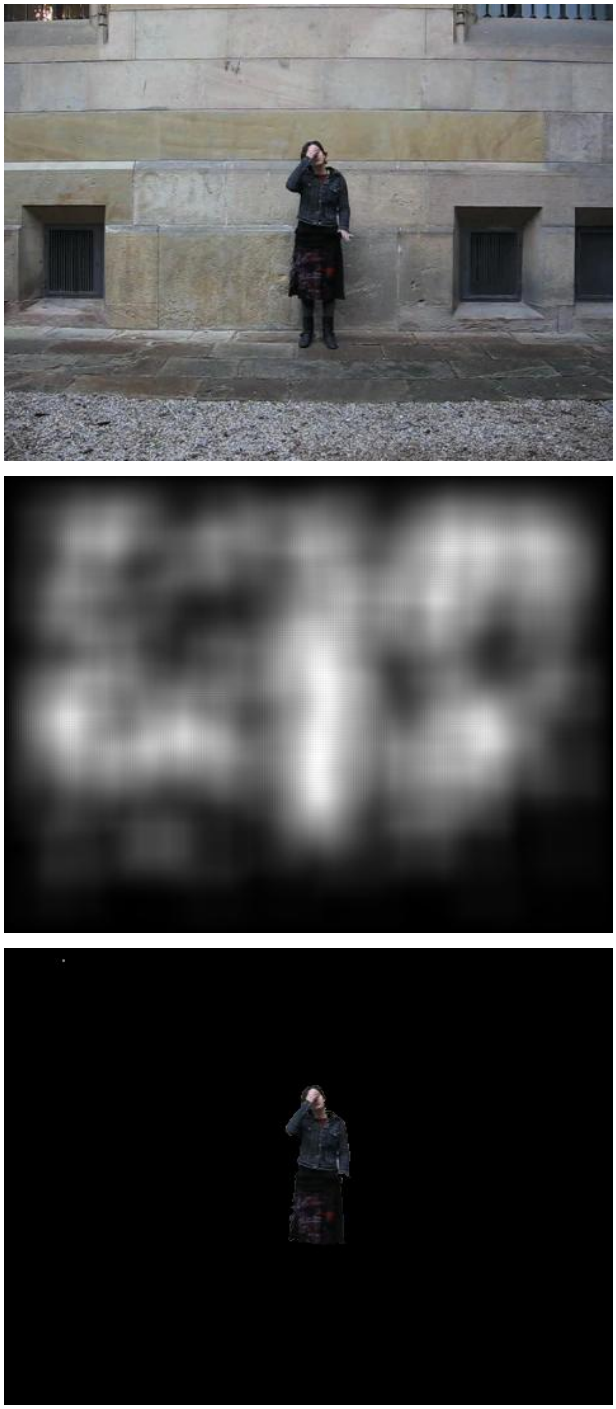


Figura 29. Resultado *grabCut* 9

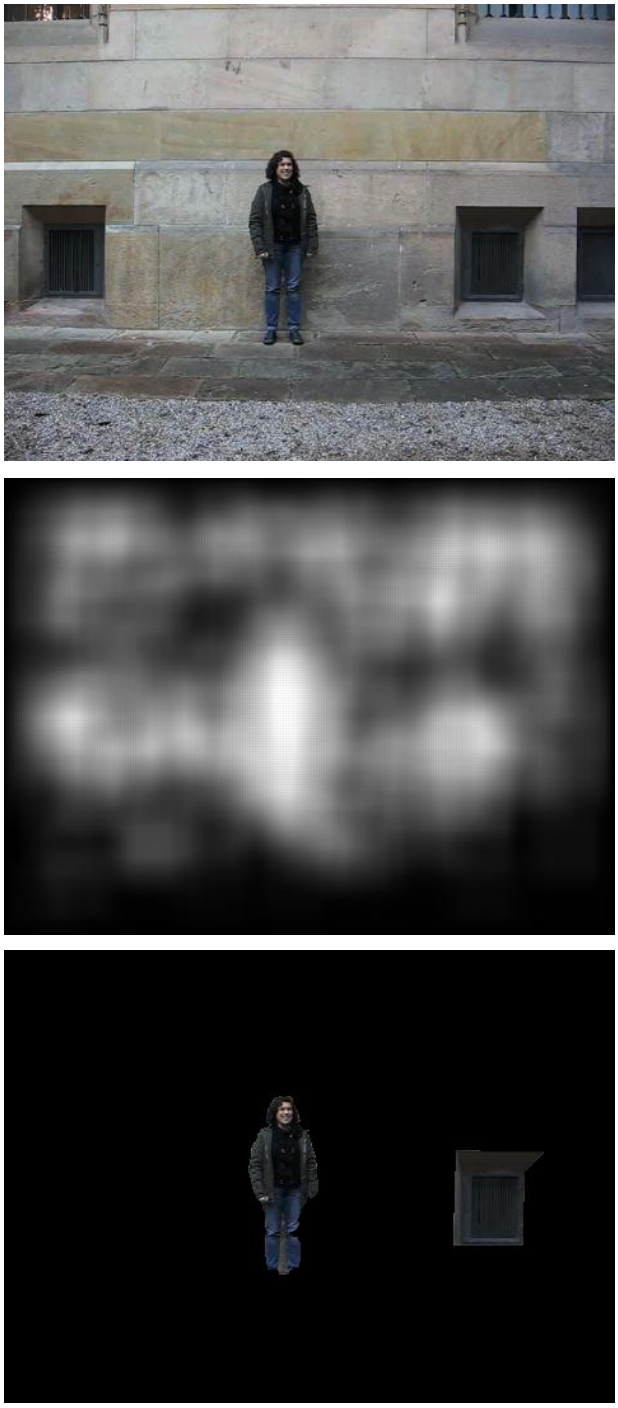


Figura 30. Resultado *grabCut* 10

Tal como se puede apreciar en la mayoría de imágenes, el nivel de segmentación persona-fondo es bastante aceptable ya que prácticamente segmenta todo el cuerpo. Las manos y los pies no han sido entrenados, motivo por el que *grabCut* no lo segmenta. Estos casos se pueden contemplar en la tercera y séptima imagen (viendo las imágenes de izquierda a derecha y de arriba abajo).

Los resultados suelen tener huecos entre las piernas, brazos en los que no se ha podido segmentar. Esto es debido a que con el actual algoritmo y la manera de funciona hace imposible segmentar entre las piernas o entre los brazos.

Además, tenemos el caso de la octava imagen, la cual muestra la segmentación de una rejilla-fondo-persona. Uno de los motivos por lo que ocurre esto es la forma de los contornos. *grabCut*, se va expandiendo de fuera hacia adentro siguiendo la regla de que si el pixel actual es igual al siguiente pixel, entonces avanzará, en caso contrario no expande. Como se puede apreciar, los primeros contornos de la rejilla difieren totalmente de la pared.

Uno de los resultados más representativos y mejor logrados, ha sido la segmentación de más de una persona en una imagen, tal como se aprecia en las dos imágenes donde aparecen dos persona prácticamente segmentadas respecto el fondo. En ambos casos las cabezas están más segmentadas de lo que deberían pero el resultado en general es bastante aceptable. Por otro lado, queda la rejilla detrás de la persona, cuyos contornos quedan siendo segmentados pero como la persona queda delante y ésta tiene unos contornos diferentes al fondo, se segmenta sin problemas.

Por otro lado encontramos los mapas de probabilidad en escala de grises generados antes de utilizar el método. En éstos se puede apreciar la figura de la persona representada como una figura bastante blanca. El procedimiento para generar estas imágenes es el explicado en el apartado anterior.

4.1.3 Resultados cuantitativos

En los diferentes resultados y teniendo en cuenta lo explicado anteriormente sobre el parámetro *PROB*, este valor ha diferido varias veces para obtener tales imágenes. Si se cuentan las imágenes como se ha dicho antes, estos son los valores de *PROB*:

Imagen	1	2	3	4	5	6	7	8	9	10
PROB	0.83	0.79	0.72	0.64	0.72	0.70	0.56	0.84	0.82	0.48

Figura 31. Probabilidades de *grabCut.*

En la tabla de arriba se puede apreciar que los valores de *PROB* varían en función de la imagen. En caso de mirar los píxeles del mapa de probabilidad de la imagen 8, la aplicación lo que haría, sería asignar todos los píxeles que sean mayores de 0.10 y menores de 0.84 a una probabilidad de ser fondo. Por otro lado, si los valores son mayores que 0.84, se asigna a una probabilidad de ser persona. Esta interpretación se puede hacer para cada con su respectivo *PROB*.

Por otro lado, se pueden llegar a obtener unos resultados menos aceptables si el valor de *PROB* cambia. A continuación se muestran dos imágenes donde la primera es obtenida al utilizar 0.60 como PROB y la segunda utiliza 0.82.

Figura 32. Resultado *grabCut* óptimo

Figura 33. Resultado *grabCut* no óptimo.

Como se puede apreciar, en la segunda imagen una parte de la cara y piernas no aparece en el conjunto de la persona. Esto es debido a que la probabilidad de esos valores son menores que 0.82 y los interpreta con probabilidad de ser fondo.

4.2 Multi-label alpha-beta-swap graph cuts

A continuación se detallan los parámetros utilizados y los resultados obtenidos.

4.2.1 Parámetros

Para la implementación del método *multi-label* se han tenido que configurar ciertos valores para que los resultados fueran lo suficientemente aceptables. Igualmente, también se detallarán algunos parámetros que dan como resultado imágenes poco aceptables para después comparar.

A continuación se explican los parámetros utilizados:

1. Número de etiquetas: 7

2. Número total de píxeles: (anchura x altura) de la imagen original.

3. Imagen original.de la base de datos.

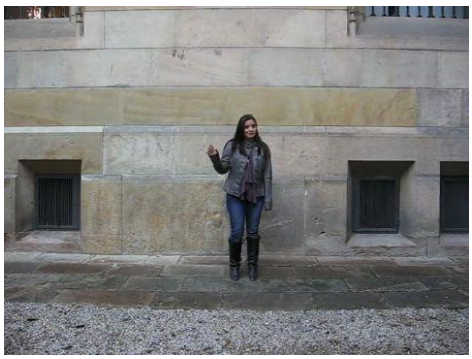


Figura 34. Imagen original.

4. La matriz *smoothCost* varía en función del número de etiquetas. Esta matriz codifica los costes de asignar una etiqueta en función de las etiquetas vecinas. En este caso se considera que todas las clases son igual de probables. Para 7 etiquetas será esta matriz:

Cabeza	Torso	Antebrazo	Brazo	Muslo	Pierna	Fondo
0	1	1	1	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	0	1	1
1	1	1	1	1	0	1
1	1	1	1	1	1	0

Figura 35. Matriz smooth cost.

5. λ: [1, 10, 100, 1000] utilizado para generar las potencias de vecindad.

6. Imagen de probabilidad para cada extremidad.

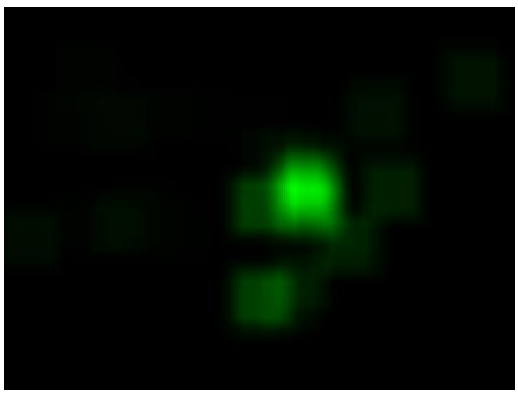


Figura 36. Imagen de probabilidad.

7. Imagen *grabCut*

Figura 37. Imagen *grabCut*.

8. Máscaras de la base de datos

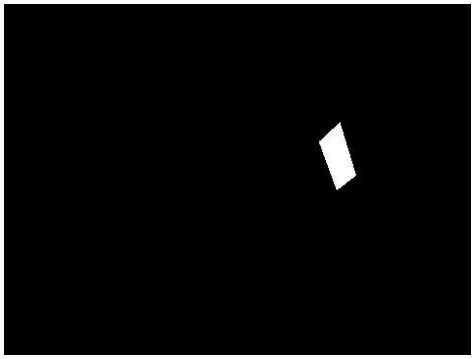


Figura 38. Máscara de ejemplo.

4.2.2 Resultados cualitativos

Una vez aplicado el método *multi-label* se obtiene las siguientes imágenes. Las primeras 6 imágenes son los mapas de probabilidad y las otras dos son la imagen original y la imagen obtenida por *multi-label*.

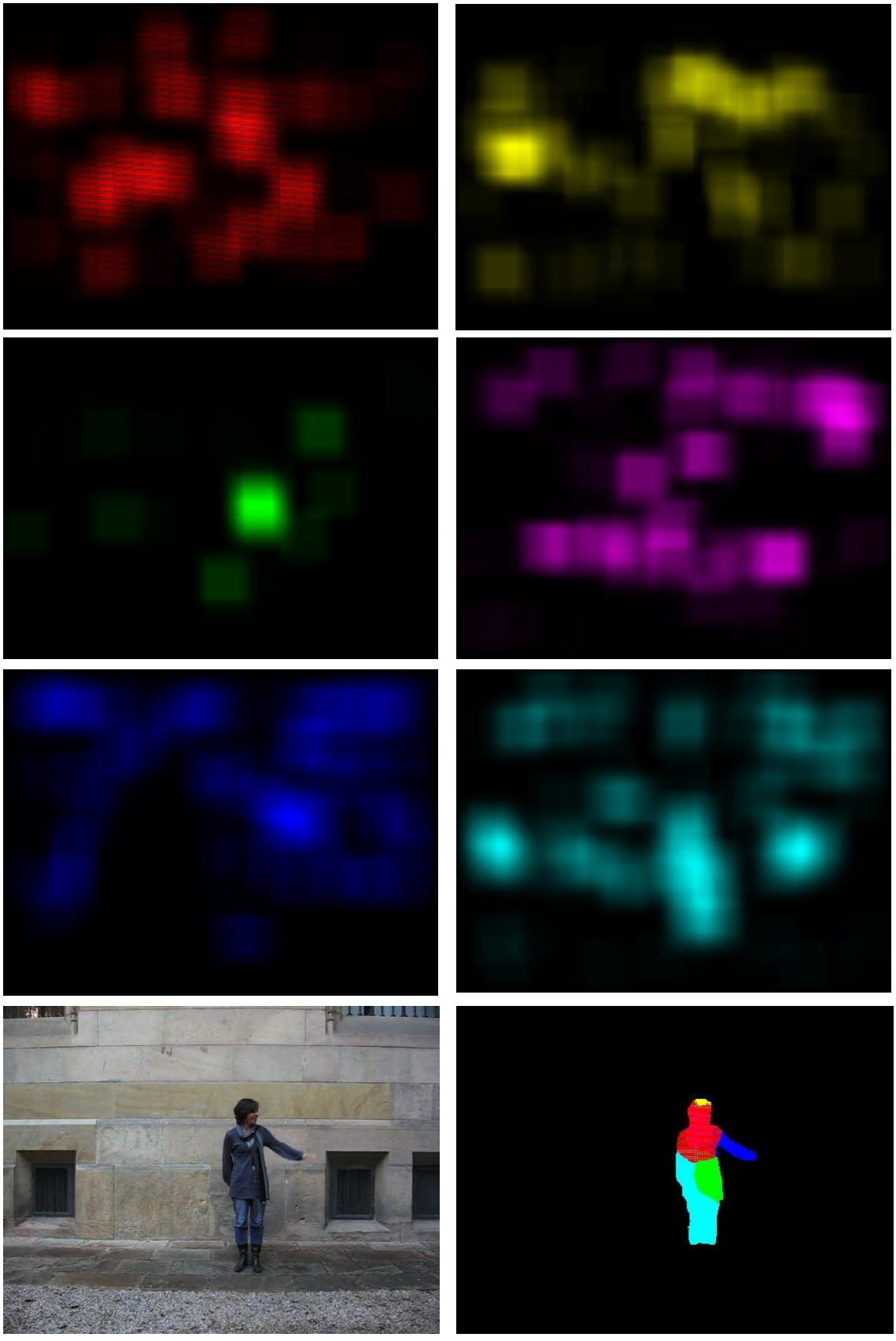


Figura 39. Resultado *multi-label* 1.

Figura 40. Resultado *multi-label* 2.

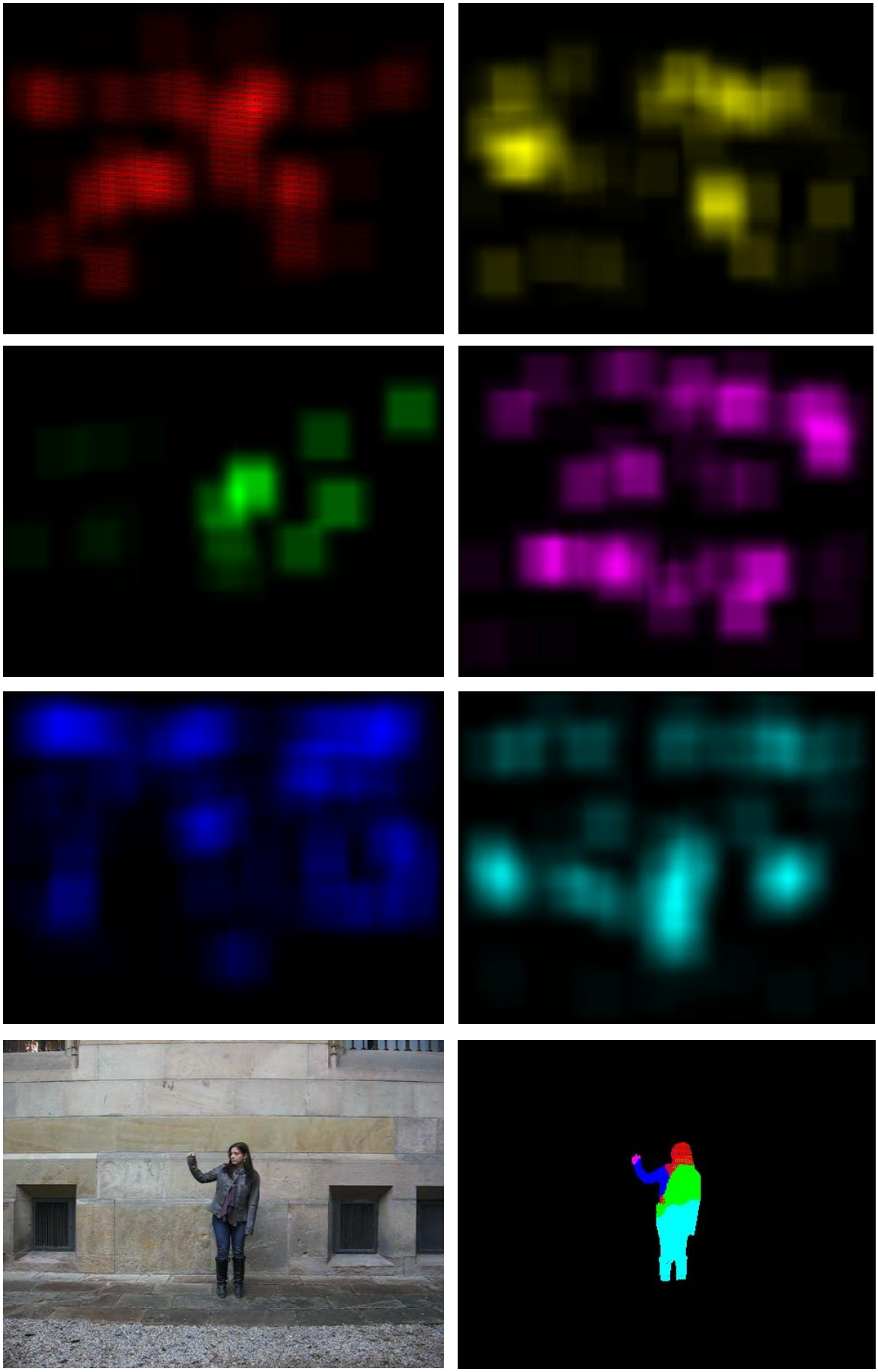


Figura 41. Resultado *multi-label* 3.

Figura 42. Resultado *multi-label* 4.

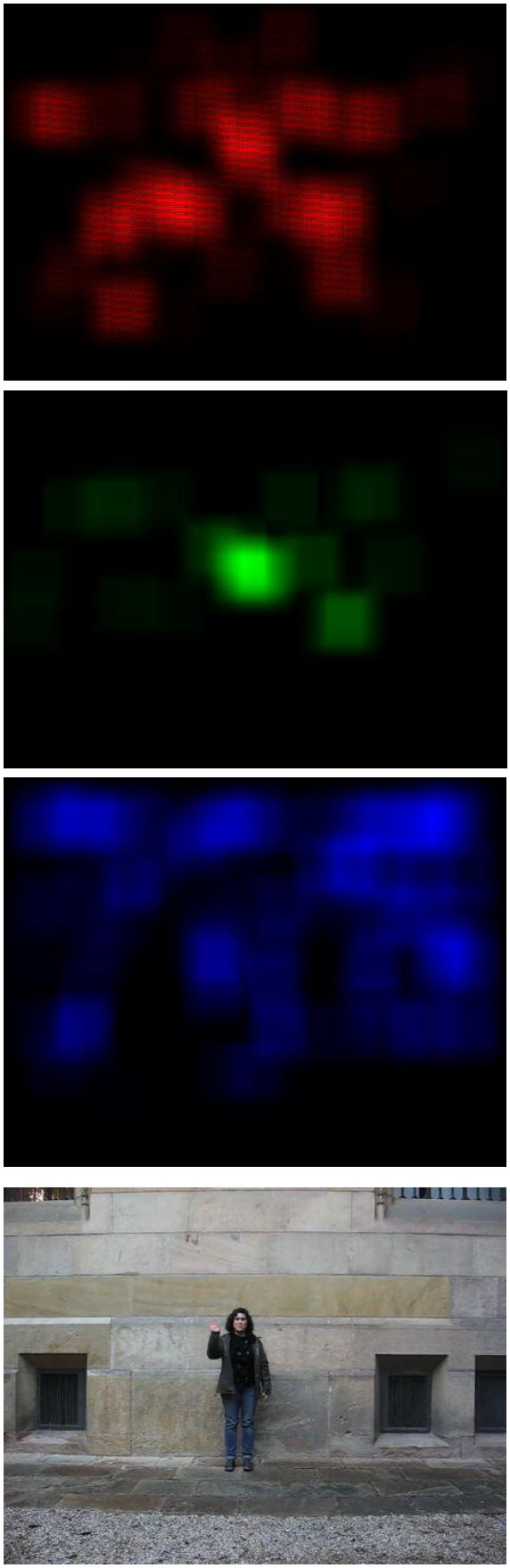


Figura 43. Resultado *multi-label* 5.

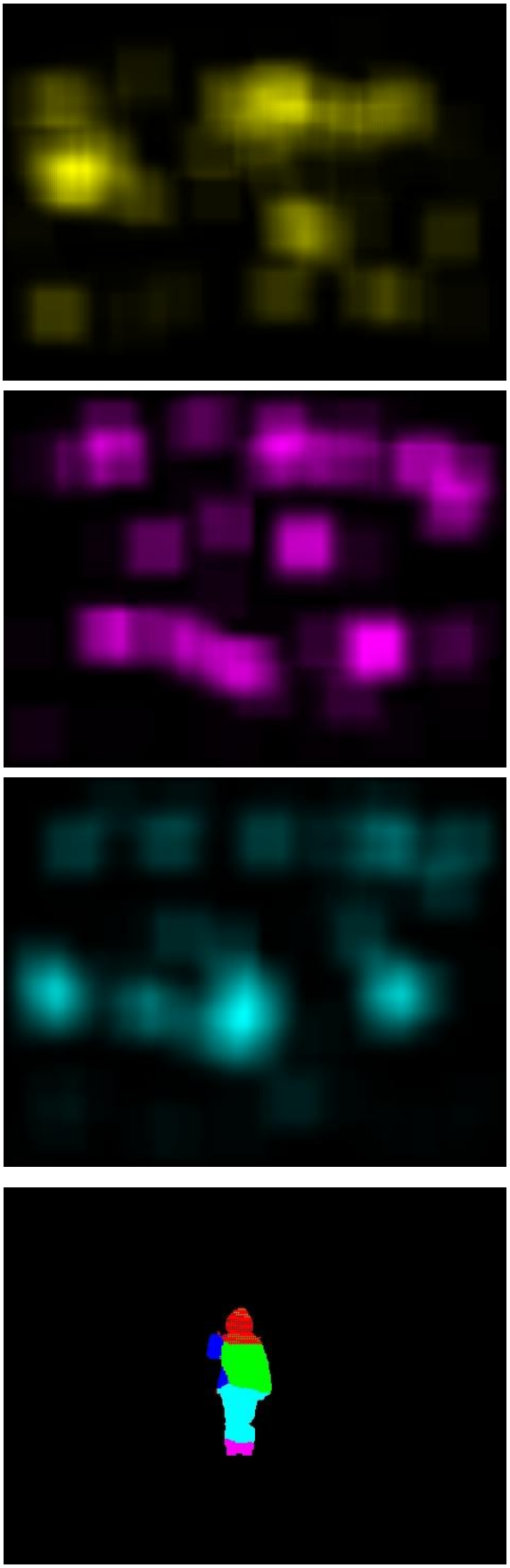


Figura 44. Resultado *multi-label* 6.

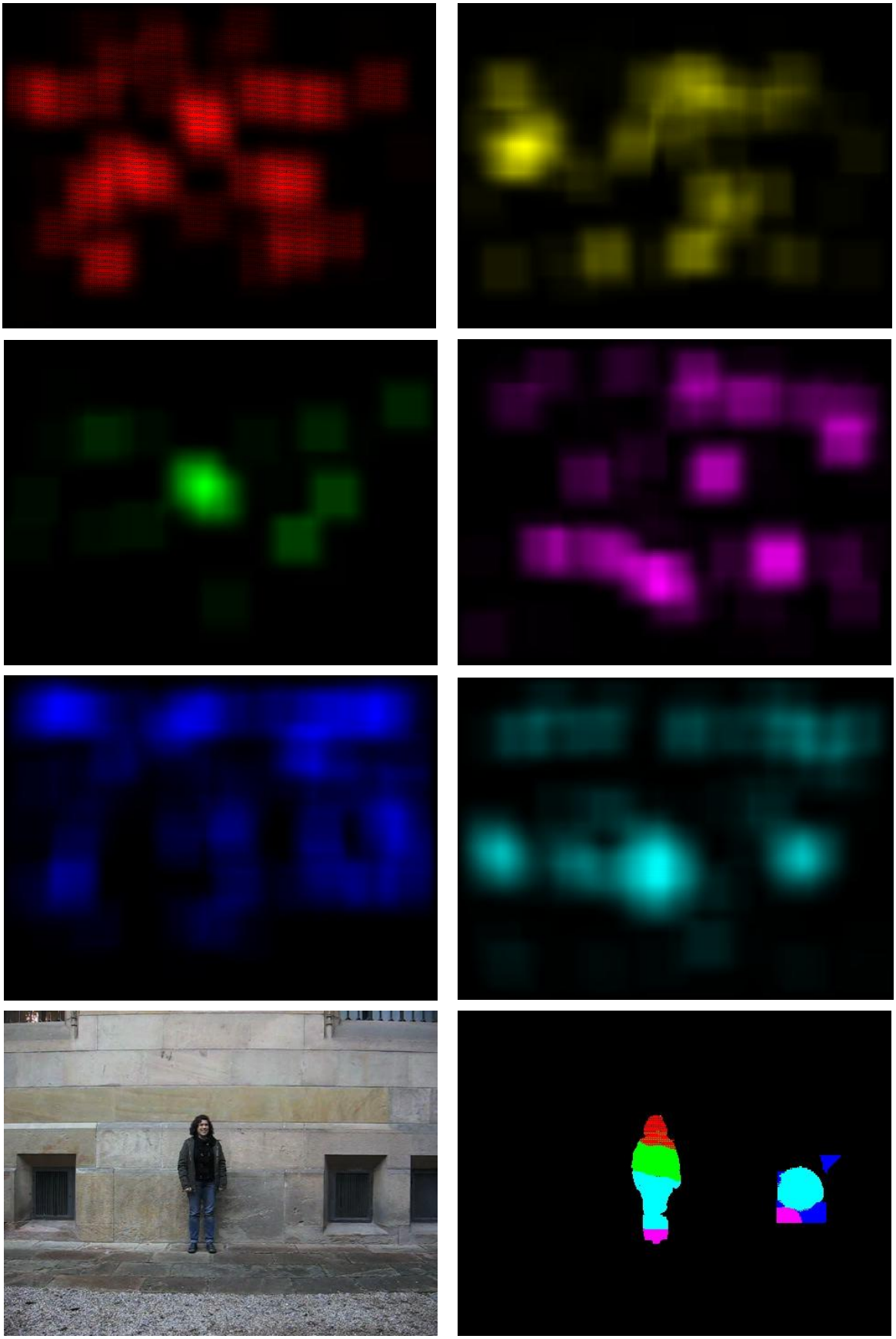


Figura 45. Resultado *multi-label* 7.

Figura 46. Resultado *multi-label* 8.

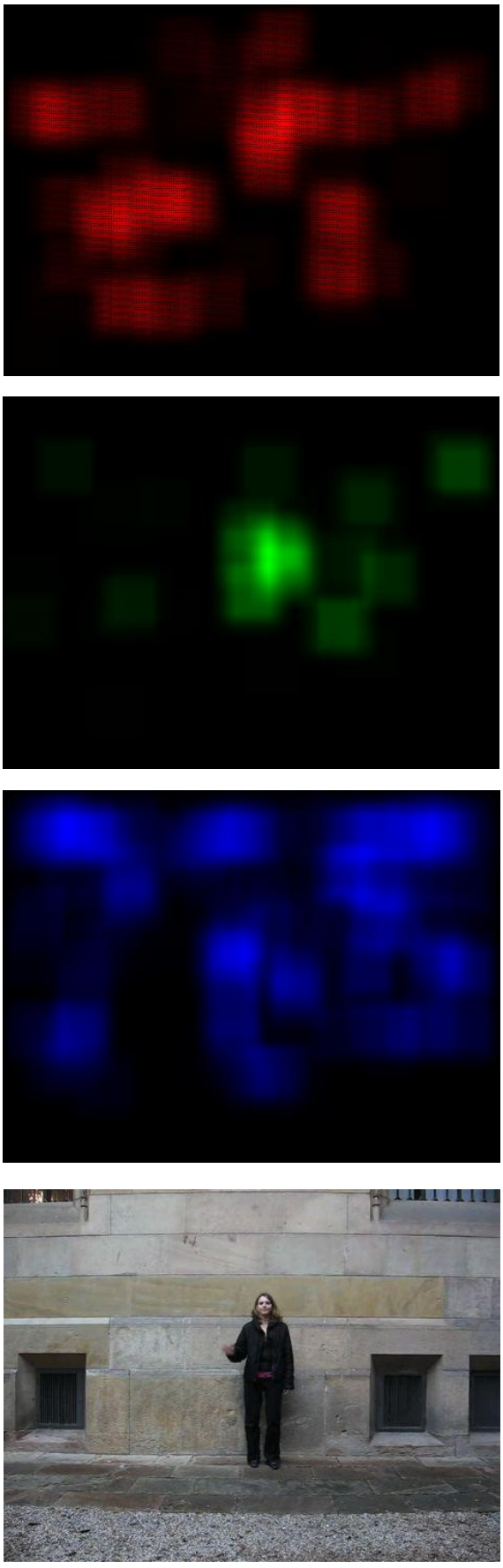


Figura 47. Resultado *multi-label* 9.

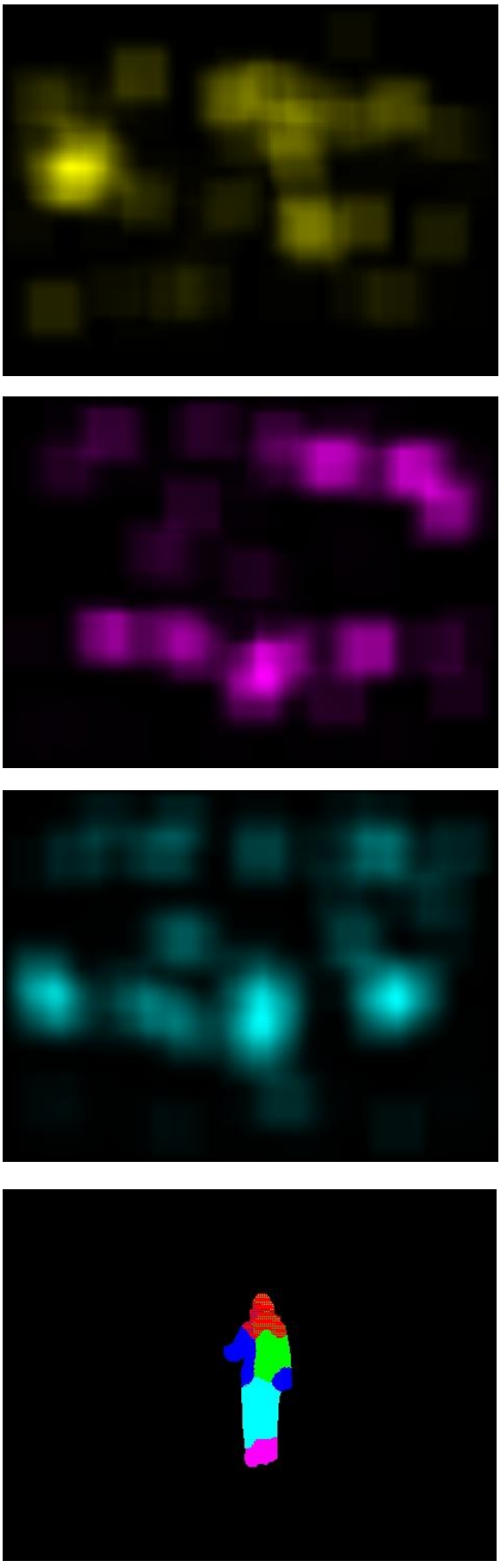


Figura 48. Resultado *multi-label* 10.

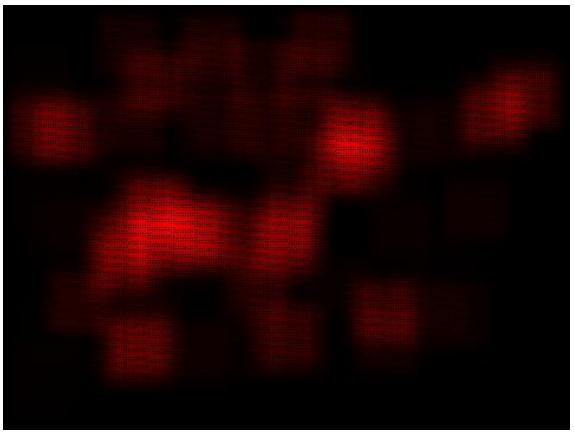
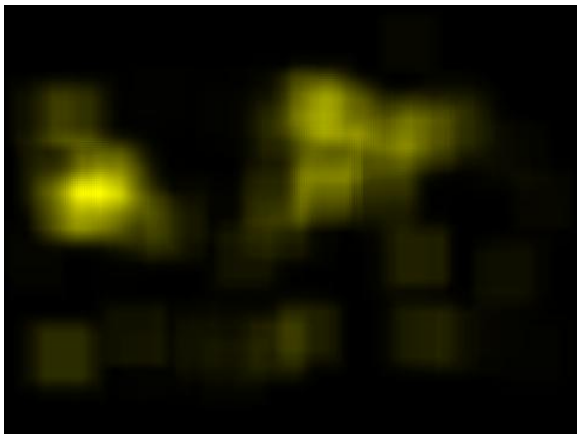
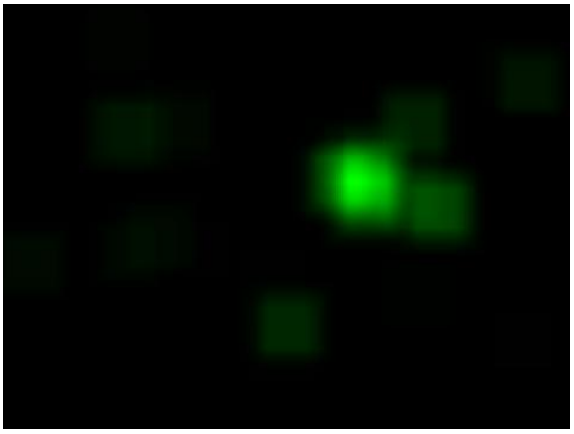
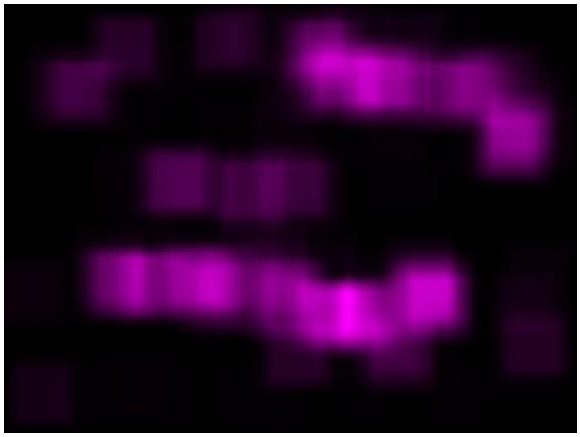
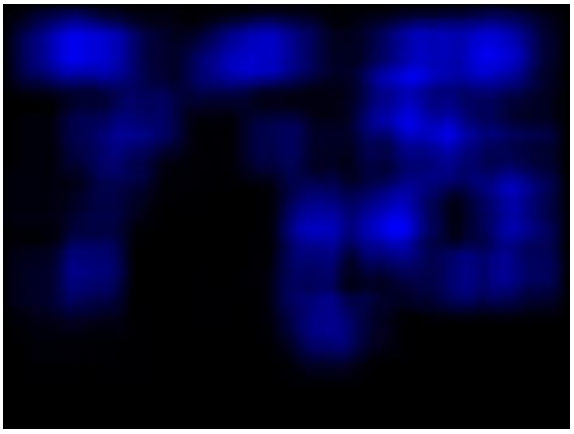


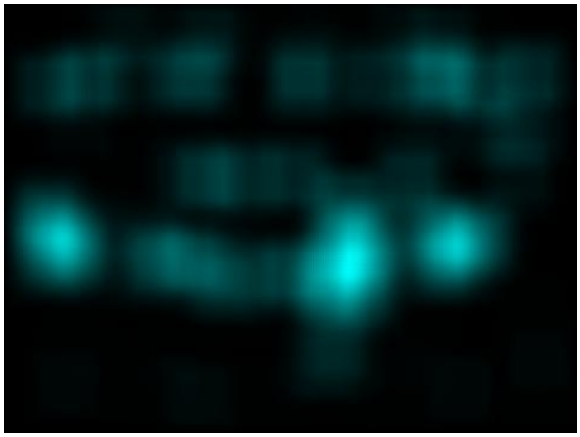
Figura 49. Resultado *multi-label* 11.











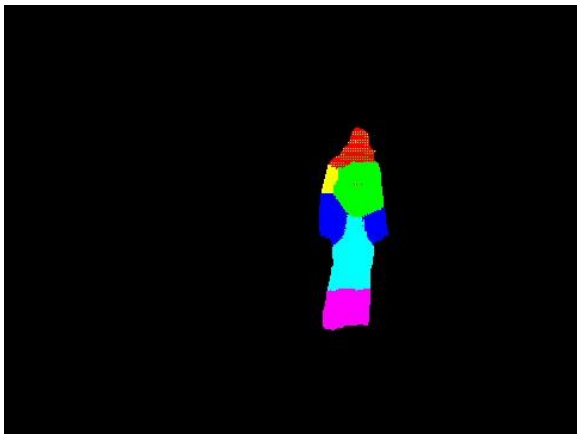


Figura 50. Resultado *multi-label* 12.

Los colores y las extremidades tienen la siguiente relación:

- rojo: cabeza
- verde: torso
- amarillo: antebrazos
- azul: brazos
- violeta: piernas
- cian: muslos

Los resultados en general son aceptables ya que si se mira extremidad a extremidad se podrá comprobar que los colores corresponden a su extremidad:

- En el caso de la cabeza, suele dibujarse ésta y parte del cuello. Aunque no tiene un color tan intenso, es decir, no hay tanta cantidad de píxeles dibujado como en el torso.

- El torso en general está bastante logrado aunque en algunas imágenes tapa la parte del antebrazo y brazo entero. Por otro lado, el número de píxeles pintados es bastante alto ya que la intensidad lo refleja.

- Los antebrazos cuestan más ya que son muy parecidos a los brazos y este tipo de zonas tubulares siempre son más costosas. En la última imagen, se puede ver el antebrazo derecho pero el izquierdo no. Una de las razones es porque el derecho está más expuesto que el otro a la cámara.

- Los brazos tienen mejores resultados que los antebrazos y en algunos casos se dibuja como antebrazo por motivos de semejanza en la forma.

- Los muslos en general tienen buenos resultados pero en algunas imágenes son dibujados como parte del torso.

- Las piernas no tienen los mismos resultados que los muslos aparte de empezar un poco por debajo de las piernas reales.

En el caso de la cuarta imagen, se puede ver que pasa prácticamente igual que en la segmentación binaria pero en este caso es segmentación de más de 2 clases. La rejilla, al tener contornos diferentes que el entorno, es segmentada de forma particular como: muslos, piernas y brazos.
Los mapas de probabilidad utilizados son uno por extremidad, tal como se visualiza arriba. Estos mapas son parecidos a los del *grabCut* y su implementación únicamente varía en que hay seis matrices, uno por cada extremidad, y no una que representa la persona.

Al aplicar el *sliding window* si la región detectada es una extremidad, por ejemplo, cabeza, la aplicación realizará la votación mencionada en el apartado *grabCut* en la matriz cabeza.

4.2.3 Resultados cuantitativos

En este apartado se mostrarán los porcentajes de acierto de las extremidades por separado, su respectiva gráfica y los aciertos agrupados por partes del cuerpo: alta, media y baja, acompañado de otra gráfica. Para realizar estas pruebas, se han utilizado 86 imágenes para realizar los cálculos.

Para calcular el porcentaje de aciertos de cada extremidad se ha calculado la intersección y la unión de los píxeles:

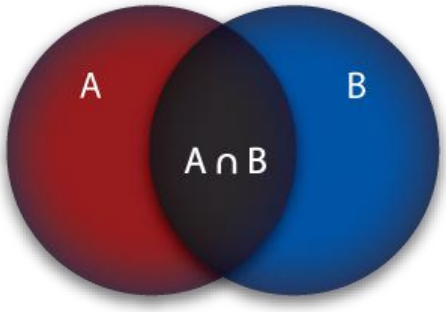


Figura 51. Intersección de dos conjuntos.

- Intersección: Los píxeles que pertenezcan a la intersección serán aquellos que sean generados por el algoritmo *multi-label* y que pertenezcan también al *ground-truth* de la máscara utilizada.

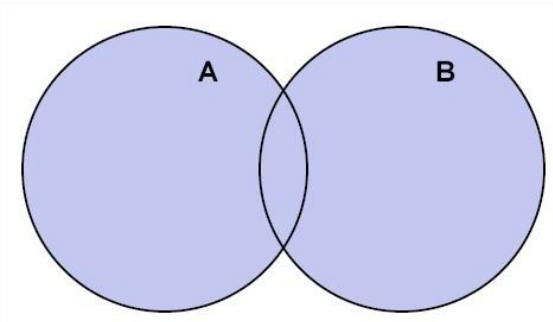


Figura 52. Unión de dos conjuntos.

- Unión: Es la unión de los píxeles del *ground-truth* de la máscara y de los píxeles detectados por el algoritmo para su correspondiente extremidad.

A continuación se muestran los porcentajes para cada extremidad:

Cabeza	Torso	Antebrazo	Brazo	Muslo	Pierna
36 %	31 %	8 %	0.04 %	0.4 %	11 %

Figura 53. Tabla de aciertos para cada extremidad.

Los resultados en general no han sido los esperados ya que los mapas de probabilidad obtenidos no han sido óptimos. Por un lado tenemos cabeza, torso y pierna que dan unos resultados discretos mientras que antebrazo, brazo y muslo apenas ponderan. En cambio, si se visualizan las imágenes parece que los resultados deberían ser mejores.

A continuación se muestra su correspondiente gráfica:

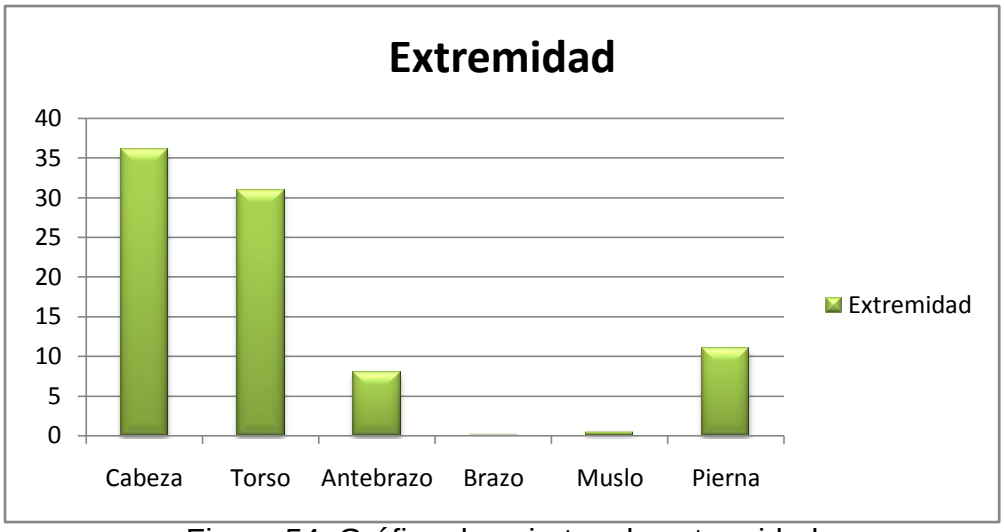


Figura 54. Gráfica de aciertos de extremidades.

La gráfica muestra resultados dispares, mientras que por un lado tenemos cabeza, torso y pierna los que más aciertos tienen, brazos y muslos apenas tienen resultados para poder visualizar sus respectivas barras.

La siguiente tabla muestra los porcentajes de acierto de los grupos de extremidades:

Parte superior (cabeza)	Parte central (antebrazos, brazos, torso)	Parte inferior (muslos, piernas)
36 %	39,04 %	11,40 %

Figura 55. Tabla de aciertos para cada grupo de extremidad.

Se puede concluir que la parte central es la que mejor detecciones y representación gráfica tiene. Varios motivos pueden ser como el tamaño en conjunto de esas extremidades como la forma. Mientras que la parte inferior es más difícil detectar ya que los muslos y piernas al ser extremidades tubulares se parecen mucho, la parte superior es solo cabeza y el tamaño no tiene las dimensiones del torso y los rasgos faciales siempre son más complicados de detectar y segmentar.

Su respectiva gráfica es la siguiente:

Figura 56. Gráfica de aciertos de los grupos de extremidad.

Como se puede apreciar en la gráfica, la parte central es la que mejor resultados obtiene seguido de la superior y la inferior.

5. CONCLUSIONES

En este proyecto, se ha desarrollado un *software* para aplicarlo a un conjunto de imágenes donde aparecen personas realizando gestos. El objetivo del proyecto es segmentar automáticamente un conjunto de extremidades de los sujetos. Primero se generaban todas las posibles regiones de cada imagen para enviarlas a las cascadas de clasificadores y obtener una detección por extremidad. Esta detección definía si era una extremidad u otra. Una vez se obtenía los mapas de probabilidad tanto de la persona como de cada extremidad, se aplicaban las técnicas *grabCut y graphCut multi-label alfa-beta-swap.*

En la implementación se ha utilizado el lenguaje C++ con librerías *OpenCV* para facilitar el tratamiento de imágenes y los diversos algoritmos de visión por computador.

Primero de todo se ha podido obtener el algoritmo *grabCut* de las propias librerías mencionadas además de varios ejemplos ya realizados para tenerlos como base para la implementación.

Además, se ha obtenido unas librerías escritas en C++ para implementar *graphCut multi-label alfa-beta-swap* para la extensión de la detección binaria a la multi-segmentación de extremidades.

Además, se ha obtenido unas librerías escritas en C++ para implementar *graphCut multi-label alfa-beta-swap* pero no ha sido tan intuitivo como la segmentación binaria, ya que en este algoritmo entran en juego varios parámetros a configurar y su utilización en el conjunto difiere bastante de la parametrización del primer método.

Una vez implementado el sistema completo se ha procedido a realizar una batería de pruebas para intentar tener unos resultados lo más aceptables posibles. En el caso de *grabCut* los resultados han sido más que aceptables ya que la segmentación final tenía una tasa de aciertos muy alta. Se ha probado para un centenar de imágenes con resultados satisfactorios.

Por otro lado, para la obtención de resultados del *multi-label* se ha demostrado que los resultados no eran los esperados y sólo se han segmentado algunas extremidades con cierto éxito. Otras extremidades han sido reconocidas con una tasa de acierto menor.

De cara a mejorar la metodología, el presente proyecto podría utilizar otro otros tipos de clasificadores para luego utilizar los algoritmos de segmentación. También, la implementación de otro tipo de generador de características de las imágenes podría tener mejoras si se aplica juntamente a otros clasificadores. Además del generador, se podrían obtener otro tipo de características para reconocer de una mejor forma las diferentes extremidades.

6. REFERENCIAS

[1] http://www.maia.ub.es/~sergio/sergio%20escalera%20homepage_006.htm

[2] http://bcnpcl.wordpress.com/research/error-correcting-output-codes/

[3] A. Hidalgo Chaparro, “Reconocimiento de Objetos Multi-clase Basado en Descriptores de Forma”, *PFC UAB, 2008*

[4] S. Escalera Guerrero, “Coding and Decoding Desing of Ecocs for Multi-class Pattern and Object Recognition”, *Tesis UAB, 2008*

[5] S. Escalera, O. Pujol, P. Radeva, “Error-Correcting Output Codes Library”, *2010*

[6] S. Escalera, O. Pujol, P. Radeva, “Loss-Weighted decoding for error-correcting output coding”, *2006*

[7] D. Reynolds, “Gaussian Mixture Models”, *MIT Lincoln Laboratory, 2000*

[8] Y. Raja, S. Gong, “Gaussian Mixture Models”, *Queen Mary and Westfield College, 1999*

[9] http://www.mathworks.es/help/toolbox/stats/bq_679x-24.html

[10] M. Saíz, “Reconstrucción tridimensional mediante visión estéreo y técnicas de optimización”, *UPC, Junio 2010*

[11] T. Collins, “Graph Cut Matching In Computer Vision”, *February 2004*

[12] C. Rother, V. Kolmogorov, A.Blake, “”GrabCut” – Interactive Foreground Extraction using Iterated Graph Cuts”, *Microsoft Reseach Cambridge, UK, 2004*

[13] C. Primo, “Segmentación interactiva multi-clase de personas”, *PFC UOC, Junio 2011*

[14] D. Martínez, “Segmentación Semiautomática de Imágenes Digitales basada en GrabCut”, *Universidad Central de Venezuela, 2009*

[15] OpenCV. http://opencv.willowgarage.com/wiki/

[16] O. Veksler, A. Delong, “Multi-label optimization”, *12 Abril 2011,* http://vision.csd.uwo.ca/code/

[17] J.C. Ortega, “Multi-clasificación discriminativa por partes mediante Códigos Correctores de Errores”, *PFC UB, Junio 2012*

7. ANEXOS

7.1 Contenido CD-ROM

El CD-ROM que se adjunta con el proyecto tiene la siguiente estructura:

./Código

- Contiene la carpeta *Test* que contiene el proyecto realizado bajo *NetBeans*. Además, se tienen que instalar las librerías *OpenCV* para luego añadirlas al *NetBeans* en el apartado configuración.

- La versión de estas librerías es la 2.3.1.

- Los archivos comprimidos son carpetas que contienen diferentes imágenes y éstas son utilizadas en la ejecución del programa. Para utilizar el código hace falta descomprimir las tres carpetas.

- El fichero *grabCut.zip* contiene las imágenes generadas con el código *Test.cpp*. Además, esta carpeta se utiliza para generar el *multi-label* con el código *graphCut.cpp.*

- El fichero *img.zip* contiene las imágenes originales utilizadas en los dos códigos comentarios en el punto anterior.

- El fichero *SVM-HOG.zip* contiene una carpeta por imagen. Ésta, contiene sus respectivas máscaras (obtenidas de la base de datos realizada por Sr. Jordi Suñé) y los mapas de probabilidad obtenidas por el clasificador *SVM*.

- El archivo *Test.cpp* y *graphCut.cpp* contienen la generación de *grabCut* y *multi-label*, respectivamente.

- La carpeta *xmlcascadas* contiene los achivos *.xml* utilizados para el reconocimiento de extremidades.

./Documentación

- Contiene el fichero Memoria.pdf. Éste detalla la explicación del proyecto.

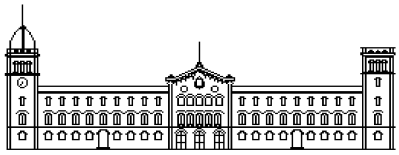
./ Imágenes

- El fichero *imágenes grabCut.zip* contiene las imágenes resultantes de aplicar el método *grabCut.*

- El fichero *Imágenes graphCut multi-label.zip* contiene las imágenes resultantes al aplicar el método *multi-label* junto un fichero *graphCut_results.txt* con la tasa de aciertos.

- El fichero *imágenes originales.zip* contiene las imágenes de la base de datos.

8. HOJA DE CUALIFICACIÓN



GRAU D'ENGINYERIA INFORMÀTICA

UNIVERSITAT DE BARCELONA

Treball fi de carrera presentat el dia de de 20
a la Facultat de Matemàtiques de la Universitat de Barcelona,
amb el següent tribunal:

Dr. President

Dr. Vocal 1

Dr. Secretari

Amb la qualificació de: