Segmentación interactiva multi-clase de personas

Carlos Primo González

junio de 2011

Supervisores

Antonio Hernández Vela
Sergio Escalera Guerrero

La segmentación de personas en imágenes es muy difícil debido a la variabilidad de las diferentes condiciones, como la postura que estas adopten, color del fondo, etc. Para realizar esta segmentación existen diferentes técnicas, que a partir de una imagen nos retornan un etiquetado indicando los diferentes objetos presentes en la imagen. El propósito de este proyecto es realizar una comparativa de las técnicas recientes que permiten hacer segmentación multietiqueta y que son semiautomáticas, concretamente en el caso de segmentación de personas. A partir de un etiquetado inicial idéntico para todos los métodos utilizados, se ha realizado un análisis de éstos, evaluando sus resultados sobre unos datos públicos, analizando 2 puntos: el nivel de interacción y la eficiencia.

La segmentació de persones es molt difícil degut a la variabilitat de les diferents condicions, com la postura que aquestes adoptin, color del fons, etc. Per realitzar aquesta segmentació existeixen diferents tècniques, que a partir d'una imatge ens retornen un etiquetat indicant els diferents objectes presents a la imatge. El propòsit d'aquest projecte és realitzar una comparativa de les tècniques recents que permeten fer segmentació multietiqueta i que son semiautomàtiques, en termes de posat de persones. A partir d'un etiquetatge inicial idèntic per a tots els mètodes utilitzats, s'ha realitzat una anàlisi d'aquests, avaluant els seus resultats sobre unes dades publiques, analitzant 2 punts: el nivell de interacció i l'eficiència.

People segmentation in images is very difficult due to the variability of different conditions, such as the position they adopt, background color, etc... To perform this segmentation, different techniques exist that, given an input image, they return a labelling of the different objects present in it. The purpose of this project is to conduct a comparison of recent techniques that allow multilabeling, and which are semi-automatic, specifically in the case of people segmentation. From an initial labeling identical for all the used methods, an analysis of them has been performed, evaluated over a public image dataset, and analyzing 2 points: interaction level, and efficiency.

Desde aquí quiero dedicar este
proyecto final de carrera a mi mujercita.
Gracias de corazón por tu infinita paciencia,
comprensión y cariño durante todo este tiempo.

Agradecimientos

Quiero agradecer en primer lugar el esfuerzo e insistencia de mis padres en que siempre siguiese estudiando. Si no hubiera sido por su ayuda, seguramente no habría podido llegar hasta aquí.

También he de darle las gracias a mi mujer, Vane, ya que ella es la que más ha sufrido mis noches de estudio, fines de semana en casita y demás. Ahora queda disfrutar de un merecido descanso.

Y por último, darle las gracias a Toni y a Sergio por la gran ayuda facilitada y dedicación durante todo este semestre. Realmente han conseguido que esté orgulloso de la realización de este proyecto y que no signifique simplemente un mero trámite para finalizar los estudios.

Contenido

1 Introducción

1.1 Contexto y Motivaciones

Actualmente, gracias a la gran potencia disponible en los computadores y la creciente cantidad de imágenes disponibles en la red, un área muy importante de la informática es la visión artificial, encargada entre muchos otros objetivos de la detección de objetos en imágenes.

Un gran problema a resolver es la detección de formas humanas en imágenes. Dentro de esta detección, además, se pueden diferenciar las diferentes partes como la cabeza, tronco superior e inferior.

Dado que existen diferentes algoritmos y aproximaciones para resolver el objetivo, he basado el proyecto en estudiar y analizar parte de estos algoritmos, estudiando las diferencias existentes entre ellos.

A esto, hay que añadir el trabajo de búsqueda de información acerca de los posibles algoritmos, conocer cómo funcionan y aprender la mejor forma de aplicarlos.

1.2 Definiciones

En el presente documento, se utilizarán términos y conceptos asociados con temas de visión por computador, por lo que a continuación se describen algunos de ellos

Visión por Computador: También conocida por visión artificial, es un subcampo de la inteligencia artificial. El propósito de la visión artificial es programar un computador para que "entienda" una escena o las características de una imagen. Los objetivos típicos de la visión artificial incluyen entre otros, la detección, segmentación, localización y reconocimiento de ciertos objetos en imágenes.

Segmentación: La segmentación en el campo de la visión artificial es el proceso de etiquetar una imagen digital en varias clases u objetos. El objetivo de la segmentación es simplificar y/o cambiar la representación de una imagen en otra más significativa y más fácil de analizar

Binaria: Dícese de la segmentación de una imagen cuando se realiza para dividirla en dos partes (objeto y fondo).

Multiclase: Se habla de segmentación multiclase o multisegmentación cuando el número de etiquetas o partes a obtener es mayor que dos.

Algoritmo de Canny: es un operador desarrollado por John F. Canny en 1986 que utiliza un algoritmo de múltiples etapas para detectar una amplia gama de contornos en imágenes.

Etiqueta: Es una de las partes segmentadas de una imagen. La eficiencia de los diferentes algoritmos se valora en función de las etiquetas que estos generan.

Semillas: Son zonas de la imagen que se marcan como pertenecientes a una etiqueta. Sirven como puntos iniciales a partir de los cuales los algoritmos intentar etiquetar la imagen por completo.

Ground Truth: En general, es el resultado correcto a un problema a resolver. En el caso de este proyecto, son imágenes segmentadas manualmente de la librería Human Limb (1).

1.3 Objetivos del proyecto

El objetivo principal de este proyecto es llevar a cabo un análisis que permita comparar los diferentes métodos o algoritmos de segmentación de imágenes en función de la misma interacción inicial, para así poder saber que algoritmo funciona mejor, generando mejores etiquetas. Para poder evaluar el resultado, se ha utilizado un conjunto de imágenes llamado Human Limb Dataset que contiene un "ground-truth", o lo que es lo mismo, una segmentación manual de cada una de las imágenes.

Para la interacción inicial existen métodos que intentan reconocer las formas como la cara o todo el cuerpo, comparando la imagen respecto a tablas que contienen múltiples estructuras faciales. Esta parte quedaría fuera del alcance de este proyecto, el cual confía en una interacción por parte del usuario en forma de mascara con varios colores, donde cada color corresponde a una etiqueta.

En las siguientes imágenes se puede observar un ejemplo. Donde la primera imagen [Imagen 1-1] es la imagen original a segmentar, y la segunda [Imagen 1-2] contiene una segmentación que servirá para evaluar la eficiencia del algoritmo

Fotografía extraída librería Human Limb[Imagen 1-1]

Segmentación manual o Ground Truth[Imagen 1-2]

1.4 Estructura de la memoria

La presente memoria esta estructura en 5 partes diferenciadas.

En la sección 2, se explica la metodología utilizada para desarrollar este proyecto, explicando que conceptos se utilizan, qué es el pre-procesado de una imagen, su segmentación y los diferentes algoritmos que se utilizan para realizar dicha segmentación.

La sección 3 explica las diferentes pruebas realizadas, que imágenes se han utilizado, cómo se ha marcado las diferentes secciones para que los algoritmos las reconozcan y los resultados obtenidos. También se explica que métrica se utiliza para evaluar los resultados y se pueden visualizar los diferentes valores obtenidos en forma de gráfica.

Finalmente, en la sección 4 se exponen las conclusiones derivadas de este trabajo, apreciando qué algoritmos funcionan mejor, y si los resultados obtenidos han tenido la suficiente calidad.

Al final de la presente memoria se encuentran los anexos, donde se incluyen datos como imágenes de resultados de ejemplo, explicación del contenido del CD-ROM, y la bibliografía y código utilizado.

2 Metodología

2.1 Pre-procesado

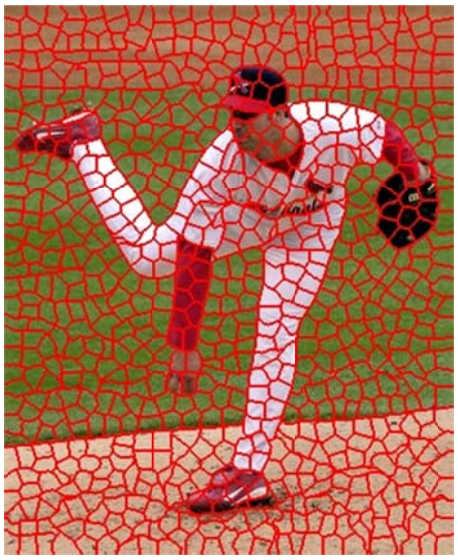
El proceso de segmentación requiere de bastantes recursos en términos de potencia de cálculo. Un buen método para facilitar este proceso es reducir el tamaño de la imagen, ya que al contener un menor número de pixeles, se reduce la cantidad de datos a tratar y el tiempo se reduce drásticamente. En función del pre-procesado, se pierde información de la imagen, por lo que es de esperar unos resultados diferentes.

La pre-segmentación consiste en reducir la imagen para que sea más pequeña y su segmentación sea más rápida, preservando la información importante a diferencia de un simple re-escalado. Existen diferentes métodos de pre-segmentación, como puede ser una reducción de la imagen de forma normal.

2.1.1 Superpixel

Existe una metodología encargada de reducir la imagen, teniendo en cuenta los contornos presentes en ésta agrupando zonas por color. Estas agrupaciones de pixeles se llaman superpixels (2) (3), donde uno de estos superpixels equivale a una agrupación de los pixeles originales.

En la imagen siguiente [Imagen 2-1] se muestra un ejemplo de superpixels, donde se puede ver como las líneas en color rojo delimitan zonas de parecido color.



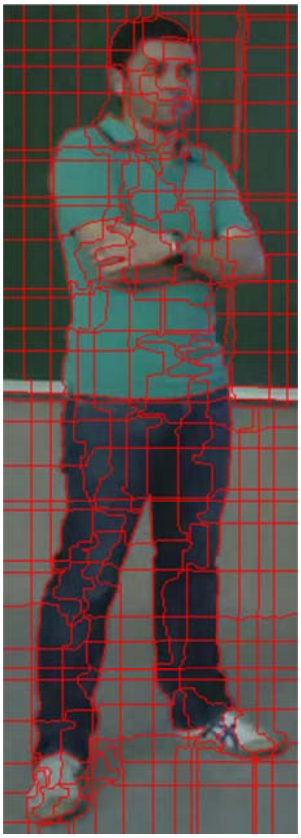
Ejemplo de superpixels (2)*[Imagen 2-1]*

Se observa como los superpixels se adaptan a los diferentes contornos de la imagen respetando la silueta del jugador, y también la separación entre la zona de césped y la tierra.

Aunque este ejemplo de superpixels es una primera aproximación bastante válida, existe la problemática de la perdida de la vecindad de los llamados superpixel. En una imagen normal, cada pixel o punto tiene 8 vecinos, 4 en horizontal y vertical, y 4 más en las diagonales. Esta estructura es la que se esperan encontrar los métodos de segmentación, y no la representación aleatoria que se observa en la imagen.

Para resolver este problema, se ha encontrado una implementación de creación de superpixels que aun realizando las agrupaciones en función de los colores y contornos de la imagen, respeta que cada superpixel tenga 8 vecinos como en una imagen normal. De esta forma, la agrupación de superpixels se puede representar en forma de imagen, y podemos aplicar directamente los métodos de segmentación como si de una imagen normal se tratara.

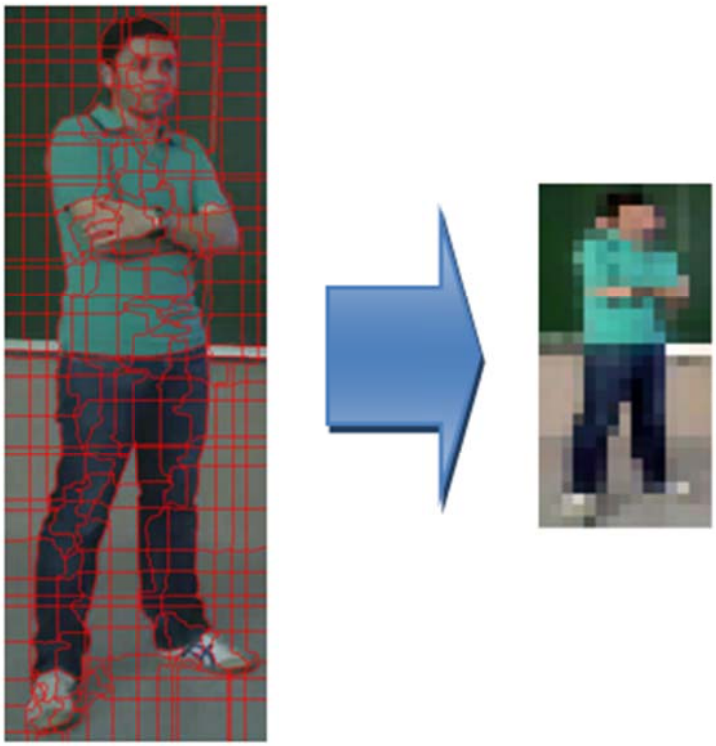
Un ejemplo de los superpixels utilizados finalmente se puede observar en la siguiente imagen



Pre-visualización superpixels en imagen [Imagen 2-2]

Como se puede observar, cada superpixel delimitado en color rojo, tiene 8 recuadros o superpixels a su alrededor. Incluso los superpixels de la cara o del pantalón respetan esta norma, aunque no lo parezca en primera instancia.

Gracias a esta propiedad, se puede representar los superpixels como una imagen. Para ello, el algoritmo de superpixels nos ha generado una matriz del mismo tamaño que la imagen original, que contiene para cada píxel un número que hace referencia al superpixel al que pertenece. Se ha desarrollado un algoritmo que a partir de la imagen original, va recorriendo uno a uno los pixeles, y agrupándolos por el código de superpixel, se calcula la media del color. Así, como se obtiene una imagen donde cada punto equivale a la media de color de la zona marcada por el superpixel, solo resta guardarla en un archivo en formato bmp (sin compresión) para que pueda ser utilizada más adelante.



Ejemplo de imagen reducida mediante superpixels [Imagen 2-3]

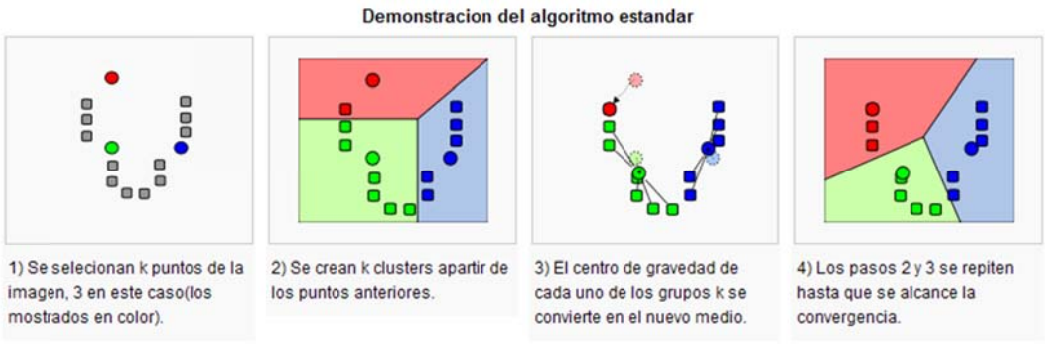
Aunque para la realización del presente proyecto no se han llegado a utilizar, existen diversos métodos de creación de "clúster" o grupos que se explican a continuación.

2.1.2 K-means

El algoritmo de las K-medias o K-means, es una técnica iterativa que se utiliza para dividir una imagen en K clústeres.

El algoritmo básico como se observa en [Imagen 2-4] es:

1. Escoger K centros de clústeres, ya sea de forma aleatoria o basándose en alguno método heurístico.
2. Asignar a cada píxel de la imagen el clúster que minimiza la varianza entre el pixel y el centro del clúster.
3. Recalcular los centros de los clústeres haciendo la media de todos los pixeles del clúster.
4. Repetir los pasos 2 y 3 hasta que se consigue la convergencia (por ejemplo, los pixeles no cambian de clústeres).



Pasos del algoritmo de k-means [Imagen 2-4]

En este caso, la varianza es la diferencia absoluta entre un píxel y el centro del clúster. La diferencia se basa típicamente en color, la intensidad, la textura, y la localización del pixel, o una combinación ponderada de estos factores.

El número K se puede seleccionar manualmente, aleatoriamente, o por una heurística. Este algoritmo garantiza la convergencia, pero puede devolver una solución que no sea óptima. La calidad de la solución depende de la serie inicial de clústeres y del valor de K.

2.1.3 Mean Shift

El algoritmo de Mean Shift (4) agrupa o crea clústeres de un conjunto de datos n-dimensional (es decir, cada punto de datos se describe mediante un vector de n valores) mediante la asociación de cada punto con un pico de probabilidad de la densidad del conjunto de datos. Para cada punto, Mean Shift calcula su pico asociado, definiendo en primer lugar una ventana esférica en el punto con radio r y calculando la media de los puntos que permanecen debajo de esa venta

El algoritmo entonces mueve la ventana a la media y se repite hasta que converge, es decir, hasta que el cambio es menor que un umbral (por ejemplo, 0.01). En cada iteración de la ventana se desplazará a una parte más densamente poblada del conjunto de datos, hasta que se llegue a un pico máximo.

Las siguientes imágenes (5) muestran el resultado de aplicar Mean Shift.

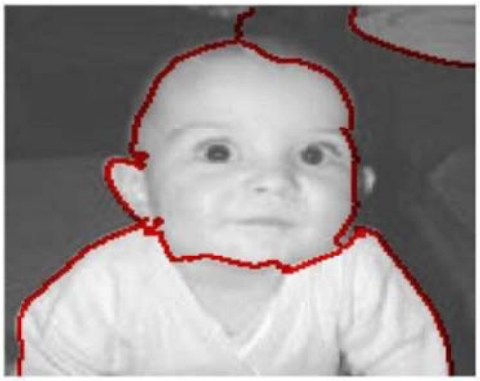
Imagen original [Imagen 2-5] *Imagen con Mean Shift aplicado [Imagen 2-6]*

La principal diferencia con K-means es que Mean Shift es un método no supervisado, es decir, no hay que indicar el número de clústeres (k)

2.1.4 Ncuts

El algoritmo Ncuts (6)realiza la segmentación de la imagen en el número de etiquetas que se le indique, pero no permite definir las semillas originales, por lo que el resultado es automático y no interactúa con el usuario. La aplicación de Ncuts como un algoritmo de pre-procesado se puede realizar de forma parecida a los métodos anteriores de pre-segmentación, el parámetro N nos permite decidir en cuántos superpixels queremos dividir la imagen.

Los resultados que generan se pueden ver en las siguientes imágenes:



Segmentación NCUTS de 5 regiones [Imagen 2-7]

Imágenes de la probabilidad de cada región en NCUTS [Imagen 2-8]

2.2 Segmentación

La segmentación en visión artificial, es el proceso de división de una imagen digital en varias partes u objetos. El objetivo de la segmentación es simplificar y/o cambiar la representación de una imagen en otra más significativa y más fácil de analizar. Más precisamente, la segmentación de la imagen es el proceso de asignación de una etiqueta a cada pixel de la imagen de forma que los píxeles que compartan la misma etiqueta también tendrán ciertas características visuales similares.

En función del número de objetos o partes en que se divide la segmentación, se puede hablar de segmentación Binaria o Multi-Segmentación.

2.2.1 Binaria

Cuando únicamente existen dos clases, se habla de segmentación binaria. Estas dos clases serán por un lado, el objeto a recortar o segmentar, y por otro lado, el fondo de la imagen.

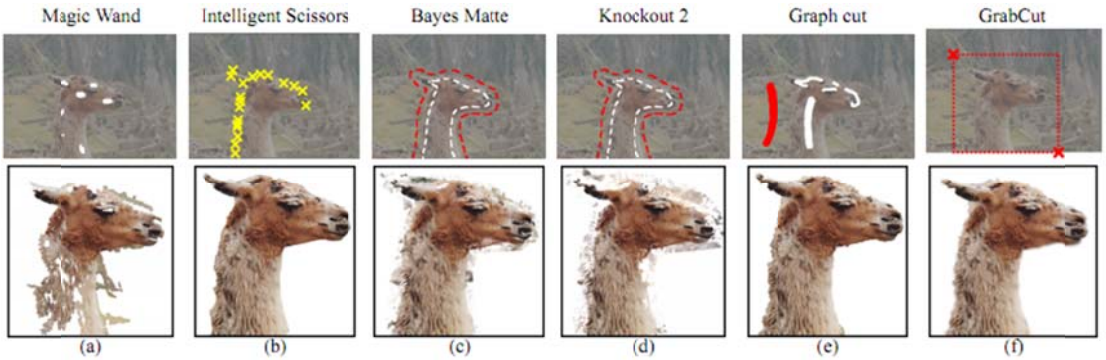
De este tipo de segmentación existen diferentes implementaciones como pueden ser la varita mágica (Magic Wand) de los programas de diseño, Bayes Matte, Knockout 2, Graph cut, GrabCut. Dado que el objetivo del proyecto era el análisis de algoritmos multi-segmentación, no entraré en explicarlos todos, sino que detallaré a continuación el modelo Grabcut.

2.2.1.1 Grabcut

Este método ha sido desarrollado por Carsten Rother, Vladimir Kolmogorov y Andrew Blake. (7)

En él se aborda el problema de la extracción eficiente e interactiva de un objeto de primer plano en un entorno complejo, cuyo fondo no puede ser eliminado de forma trivial. El objetivo es lograr un alto rendimiento con un esfuerzo modesto por parte del usuario.

Según se puede observar en la siguiente imagen [Imagen 2-9], 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.



Comparación de métodos de segmentación [Imagen 2-9]

Tal y como se observa en la imagen, la segmentación realizada por los métodos Intelligent Scissors, Graph cut y GrabCut son bastante buenas, pero es importante reseñar que en caso de grabcut el usuario casi no necesita interactuar.

En el comienzo de este proyecto, se ha utilizado este algoritmo para evaluar su calidad, eficiencia y que resultados permite generar. Realmente los resultados son bastante buenos, con una segmentación muy ajustada a los bordes de la imagen.

La implementación de este algoritmo venia codificada como ejemplo de las librerías OpenCV (8).

Dado que junto 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 más fácil y exacto evaluar los procesos, ya que ante una misma máscara de entrada, los resultados son comparables.

Su utilización se puede ver en la [Imagen 2-11], donde se ha seleccionado el área de interés, y en este caso se ha marcado el objeto de color rojo, y el fondo se ha marcado en azul. Una vez se ha marcado las zonas, se ejecuta el algoritmo y se obtiene el resultado de [Imagen 2-11]:

Imagen a segmentar con semillas aplicadas [Imagen 2-10]

Imagen ya segmentada [Imagen 2-11]

2.2.2 Multi-Segmentación

En el caso de la multi-segmentación, se busca el poder dividir la imagen original en un número de partes mayor a 2. El objetivo de este proyecto se basa en la identificación de las diferentes partes de una persona, como son la cabeza, torso superior y extremidades inferiores.

En este modo, es necesario una interacción del usuario un poco mayor, ya que por cada clase u objeto a segmentar, se necesita una marca inicial para que funcione como semilla.

2.2.2.1 α-β swap y α-Expansion

Los algoritmos α-β Swap y α-Expansion (9) utilizados en este proyecto se encargan de optimizar y segmentar la imagen realizando un etiquetado de los diferentes pixeles que la forman. Este etiquetado se realiza permitiendo que un gran número de pixeles cambien sus etiquetas simultáneamente.

En la figura siguiente [Imagen 2-12] se puede visualizar un ejemplo de estos algoritmos. 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 y cuarta imagen enseñan cómo se cambian un gran número de etiquetas simultáneamente.

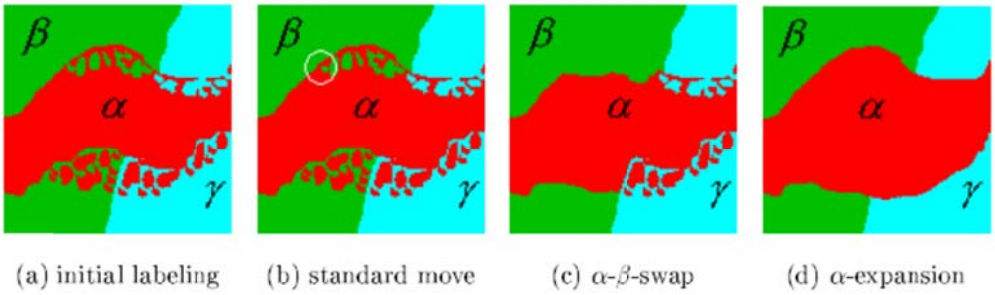


Imagen resultado de aplicación de α-β Swap y α-Expansion [Imagen 2-12]

La diferencia entre α-β Swap y α-Expansion es el método que utilizan para cambiar la etiqueta a la que pertenecen los diferentes píxeles.

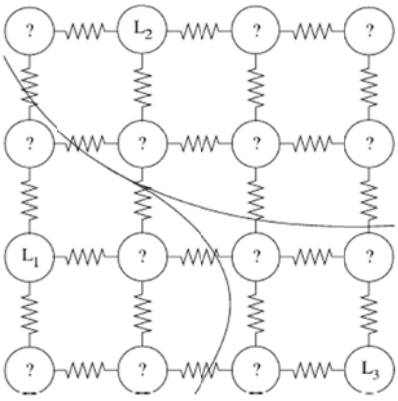
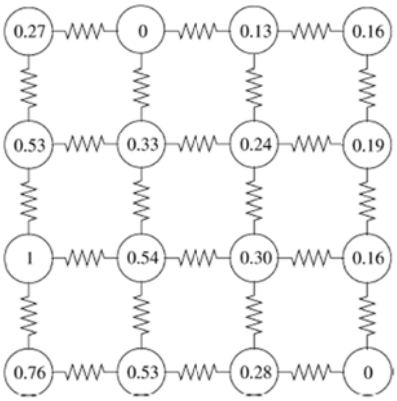
α-β Swap se caracteriza por mover algunos pixeles etiquetados α a la etiqueta β pero a su vez, algunos pixeles etiquetados como β se etiquetan como α. Es decir se intercambian (Swap) pixeles entre dos etiquetas.

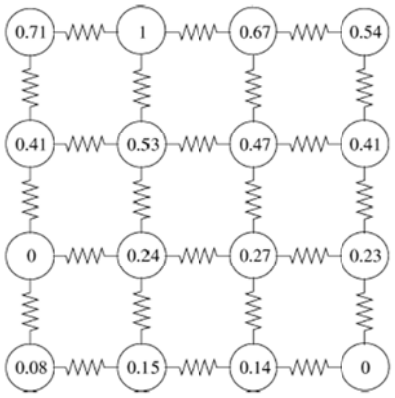
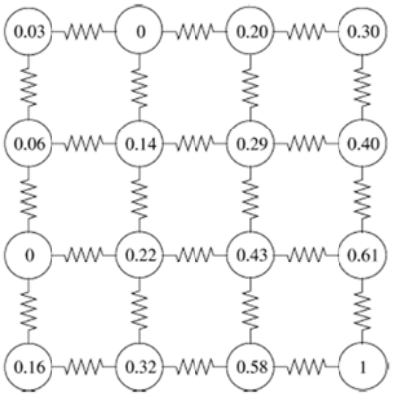
α-Expansion se encarga de mover cualquier número de pixeles, sin importar su etiqueta para marcarlos como α.

2.2.2.2 Random Walks

Random Walks (10) es otro método para realizar multi-segmentación interactiva de imágenes. Dado un pequeño número de pixeles definidos por un usuario, pertenecientes a una etiqueta u otra, se puede determinar rápida y analíticamente la probabilidad de que un pixel no etiquetado pertenezca a una etiqueta en concreto. Este proceso tiene en cuenta que a cada pixel se le asigna una probabilidad de pertenecer a cada una de las etiquetas en función de una trayectoria de pasos aleatorios. Gracias a esta asignación, es posible obtener una segmentación de la imagen de alta calidad.

El funcionamiento se puede visualizar a continuación:

En esta imagen tendríamos las semillas con la segmentación realizada. Las semillas se llaman L1, L2 y L3	Esta imagen mostraría las probabilidades de que desde cualquier nodo se alcance de forma aleatoria a la semilla L1
	
Semillas iniciales y su segmentación [Imagen 2-13]	*Probabilidades de alcanzar L1 [Imagen 2-14]*

Esta imagen mostraría las probabilidades de que desde cualquier nodo se alcance de forma aleatoria a la semilla L2  *Probabilidades de alcanzar L2 [Imagen 2-15]*	Esta imagen mostraría las probabilidades de que desde cualquier nodo se alcance de forma aleatoria a la semilla L3  *Probabilidades de alcanzar L3 [Imagen 2-16]*

3 Experimentos

Una vez definidos los objetivos de este proyecto y explicada la metodología en que se basaran las pruebas, se procede a explicar los trabajos realizados de preparación de entorno, datos que se utilizan y que resultados se han obtenido.

Los experimentos realizados se han basado en, a partir de un conjunto de imágenes junto con las imágenes "ground truth" correspondientes, ir ejecutando los diferentes métodos de forma ordenada recopilando los valores calculados de eficiencia en función de las métricas definidas.

Para realizar un análisis en el que los resultados sean comparables entre sí, se ha acotado el conjunto de métodos a aquellos que permiten realizar multi-segmentación.

También se ha estudiado los grados de interacción del usuario. Dado que para los algoritmos se utiliza la misma entrada, se simula la mayor interacción de un usuario sumando las máscaras de entrada, combinándolas.

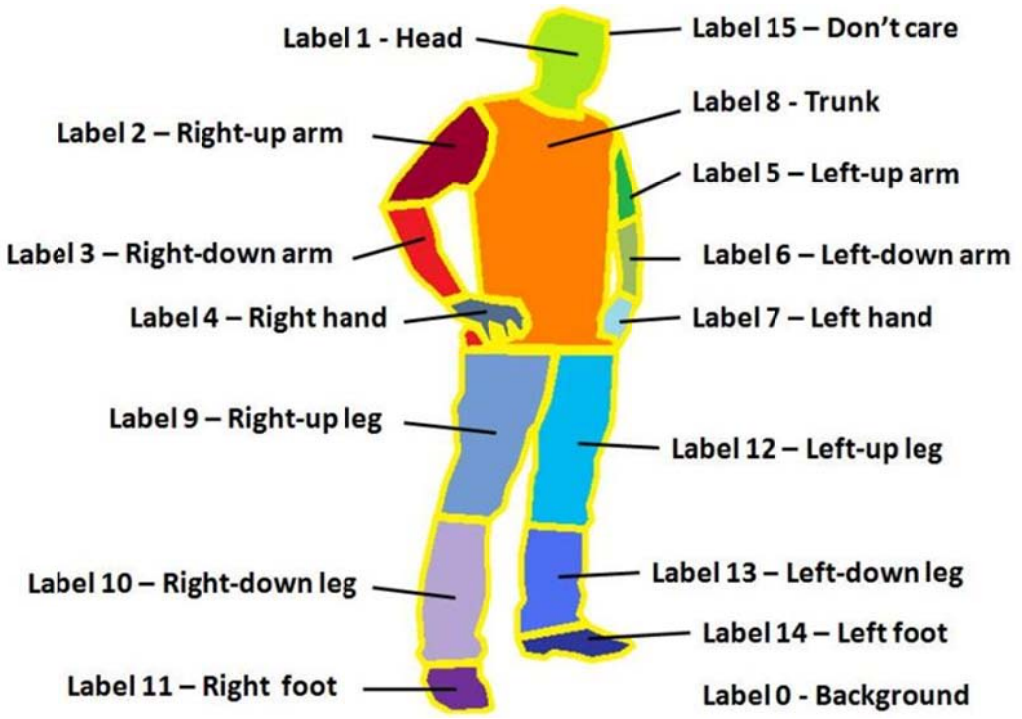
3.1 Datos de entrada al sistema

Para poder efectuar un análisis de forma que el resultado de este sea objetivo y para nada aleatorio, se ha trabajo con la biblioteca de imágenes Human Limb Data Set (1).

Este conjunto de imágenes contiene 277 fotografías de 25 personas diferentes, con fondos de diferente complejidad.

Imágenes de ejemplo de la librería Human Limb [Imagen 3-1]

Para cada una de las imágenes existe otra imagen correspondiente al ground-truth o segmentación manual, donde se ha marcado cada uno de los miembros que forman la persona.



Ejemplo de cómo están etiquetadas las fotografías de la librería Human Limb [Imagen 3-2]

Esta imagen [Imagen 3-2] tiene marcadas 14 partes diferentes, aunque para este análisis se han simplificado y agrupado en 4 partes.

La correspondencia utilizada es

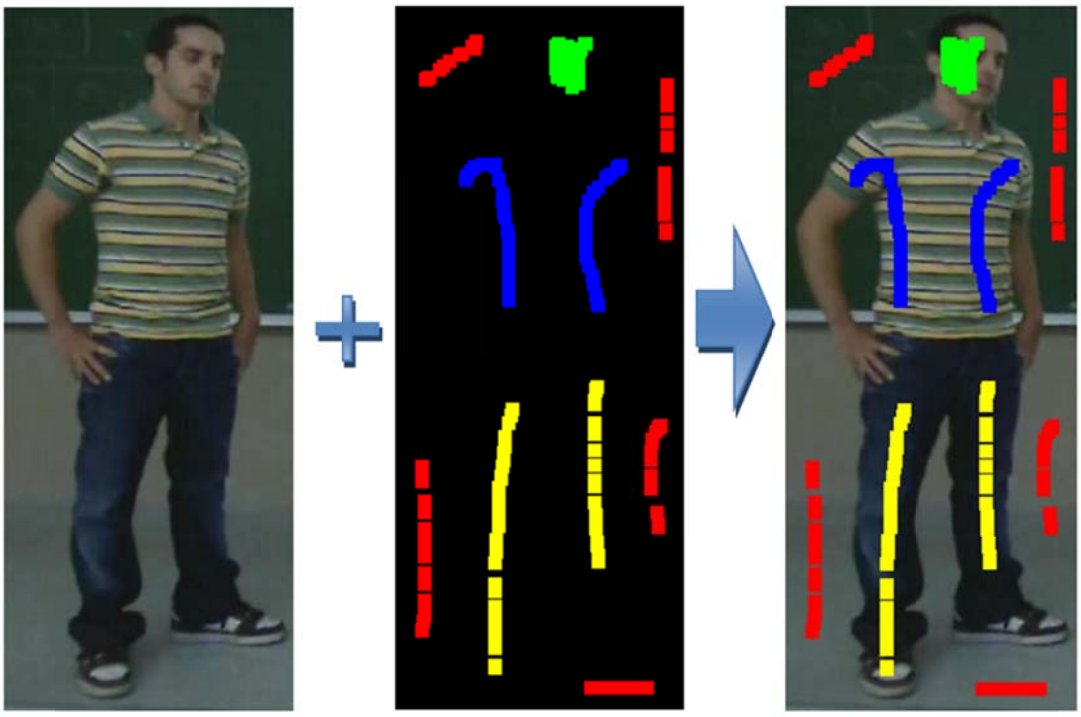
Etiquetas de las máscaras	Etiquetas Ground-Truth
Label 1 - Fondo	Label 0 - Background
Label 2 – Cabeza y manos	Label 1 – Head Label 4 – Right hand Label 7 – Left Hand
Label 3 – Tronco Superior	Label 2 – Right-up arm Label 3 – Right-down arm Label 5 – Left-up arm Label 6 – Left-down arm Label 8 – Trunk
Label 4 – Tronco inferior	Label 9 – Right-up leg Label 10 – Right-down leg Label 11 – Right foot Label 12 – Left-up leg Label 13 – Left-down leg Label 14 – Left foot
No se utiliza y es ignorada	Label 15 – Don't care

La segmentación realizada por los diferentes métodos, genera un resultado con un etiquetado de la imagen en función de las 4 etiquetas mostradas en la primera columna de la tabla. Utilizando esta correspondencia, se pueden comparar las etiquetas retornadas por los

algoritmos de segmentación, y las etiquetas manuales, para poder verificar si un algoritmo funciona mejor o peor que otro.

Los diferentes algoritmos necesitan un mínimo de etiquetas con las que comenzar la segmentación. En condiciones normales, sería el usuario el encargado de facilitarlas, marcándolas en la imagen mediante el ratón, en el momento de segmentar la imagen. En nuestro estudio, como se quiere evaluar la calidad de los métodos, se ha buscado una forma de poder pasar exactamente las mismas etiquetas iniciales a todos los algoritmos.

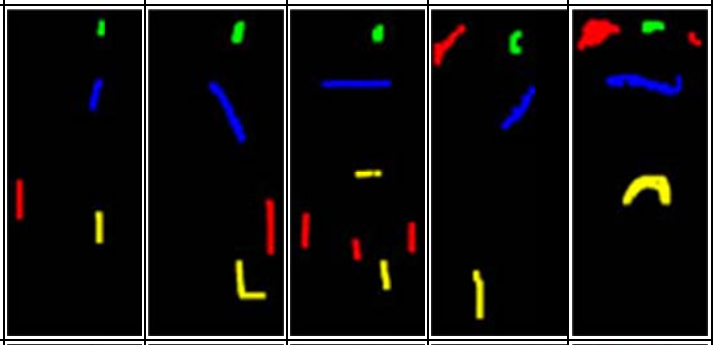
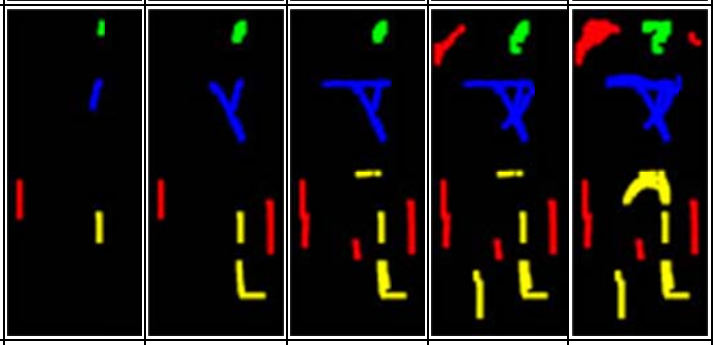
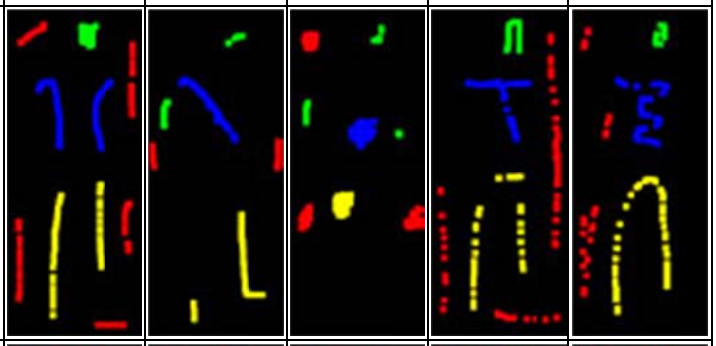
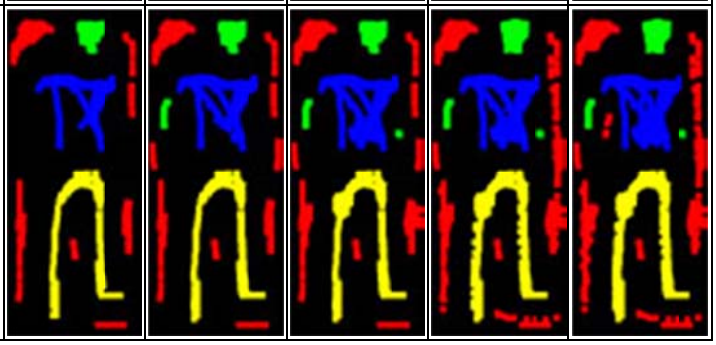
La solución ha sido utilizar unas imágenes donde cada etiqueta está asociada a un color. La siguiente imagen muestra un ejemplo:



Relación entre imágenes y máscaras utilizadas [Imagen 3-3]

El color rojo corresponde a la etiqueta fondo, la verde es la cabeza, azul para tronco superior y amarillo para el tronco inferior. El resto de píxeles que no contienen ninguna etiqueta se representan en negro. Como esta imagen se utiliza de igual forma en los diferentes algoritmos, los resultados en diferentes repeticiones siempre serán los mismos.

Para representar diferentes interacciones del usuario, se ha realizado una combinación de las máscaras. Esta combinación funciona sumando las máscaras entre sí. Por ejemplo, la máscara combinada 2 es la suma de las máscaras 1 y 2, la máscara combinada 3, es la suma de las máscaras 1, 2 y 3. De esta forma se realiza la combinación hasta la máscara número 10.

Número de máscara	1	2	3	4	5
Normal					
Combinada					
Número de máscara	6	7	8	9	10
Normal					
Combinada					

Diferencias entre las máscaras normales y combinadas [Tabla 3-1]

Este proceso de combinar las máscaras se realiza para simular el caso en particular de que un usuario va interaccionando con los algoritmos a medida que va obteniendo resultados. Se puede ver como las máscaras combinadas tienen mucho más detalle que las normales

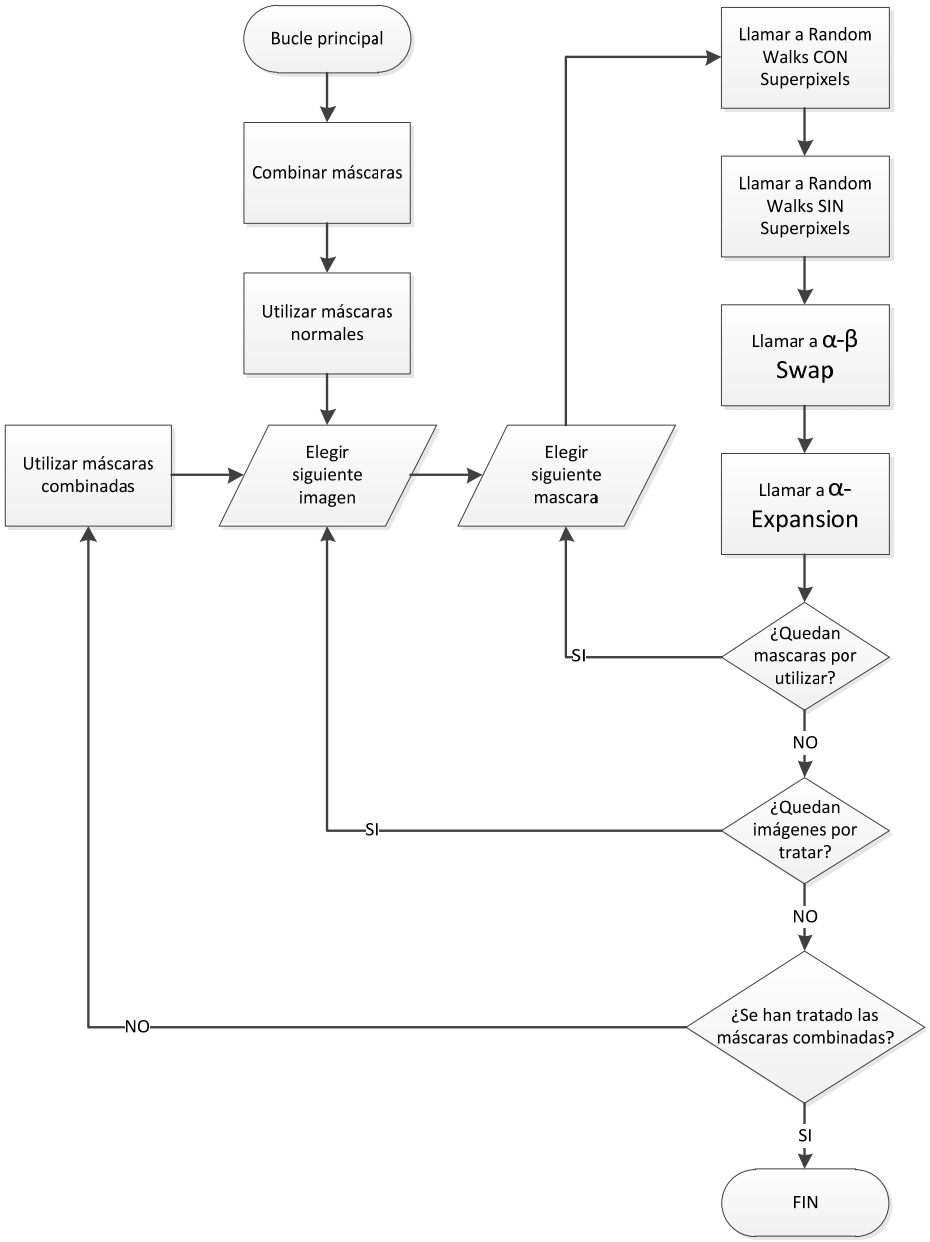
Cómo para cada imagen se han creado 10 máscaras y se ha trabajado con 10 imágenes, para facilitar la creación de estas y evitar posibles errores al marcar con un color erróneo o utilizar un tamaño de máscara diferente a la imagen, se ha creado una función codificada en Matlab que asiste al usuario en la creación de estas. El uso de esta función se puede ver en el punto 6.1.

3.2 Métodos y Parámetros

Para cumplir los objetivos de este proyecto, se han codificado diversos métodos que han permitido ejecutar todas las pruebas de una forma automática, y también se han codificado los diferentes métodos para adaptarlos a la estructura impuesta.

El desarrollo y codificación de los algoritmos se ha realizado bajo el software Matlab. Este tiene como prestaciones básicas la manipulación de matrices, representación de datos y funciones, implementación de algoritmos, creación de interfaces de usuario y comunicación con programas en otros lenguajes.

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



Este bucle comienza por un recuadro llamado combinar máscaras tal y como se explica en el punto anterior.

Una vez se ha realizado la combinación de las máscaras, se procede a escoger la primera imagen y, para cada una de sus máscaras, se llama a los 3 métodos implementados.

Los métodos y algoritmos finalmente implementados han sido 4, uno de pre-segmentación que es el de superpixel, y los otros 3 son de multi-segmentación.

3.2.1 Superpixel

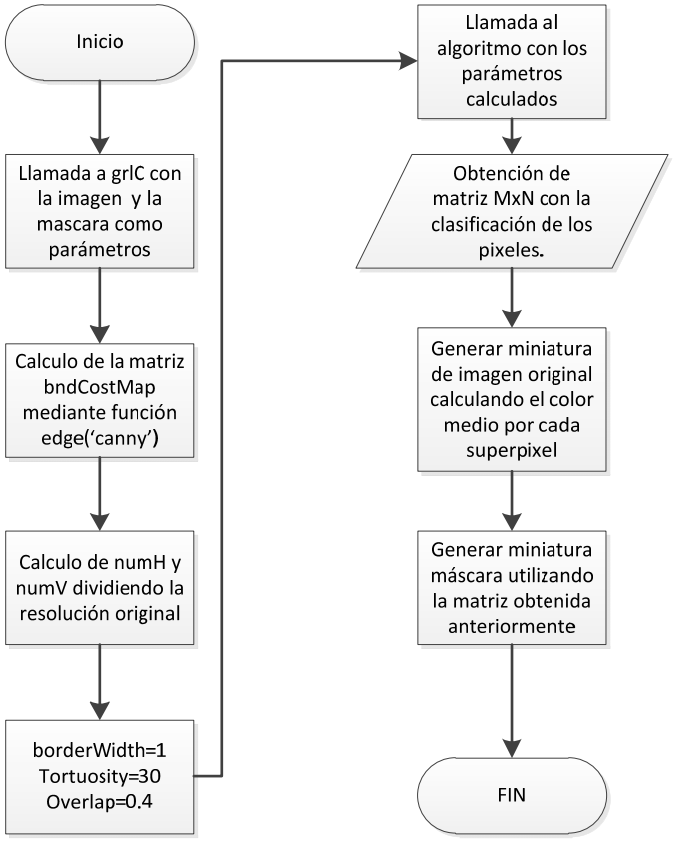
El código utilizado ha sido el llamado Superpixel Lattices (3). Dicho código se facilita en forma de archivo compilado, por lo que no se tiene acceso al código fuente del mismo, y necesita de una versión de Matlab y sistema operativo de 32 bits. Esto ha forzado a utilizar una versión de Windows de 32 bits, aunque no ha supuesto ningún problema para el resto de algoritmos.

Se utilizan dos archivos para obtener la clasificación en superpixel. El primer archivo, llamado grlC.m es el código Matlab que se encarga de preparar los parámetros para efectuar la llamada al algoritmo. Los parámetros que se utilizan son:

- bndCostMap: matriz de m x n de valores de 0 a 255 indicando los bordes o contornos de la imagen. Se calcula mediante Canny de la imagen en escala de grises.
- numH y numV: indican el número de superpixels que se generarán, tanto en horizontal como vertical. Se puede entender que será la resolución de la imagen que obtendremos. Para mantener la misma relación en las proporciones que la imagen original, los valores de numH y numV se han calculado dividiendo los píxeles de la imagen por una constante. En el caso de este estudio, se ha utilizado el número 6.
- borderWidth: Se utiliza para prevenir que diferentes caminos utilicen el mismo contorno de la imagen, normalmente se utiliza el valor 1.
- Tortuosity: Es una constante que permite definir como de tortuosos serán los caminos que separan los superpixels. Aunque el valor puede ir de 0 a 255, según los creadores, se recomienda dejarlo entre 0 y 100.
- Overlap: Constante que especifica el solapamiento entre las tiras utilizadas en la construcción de los superpixels. El valor utilizado se ha fijado en 0.4

El segundo archivo, llamado grl.mexw32, recibe estos parámetros y genera una matriz MxN donde cada elemento contiene un número indicando el superpixel al que pertenece. Esta matriz, al ser idéntica en tamaño a la imagen original, nos relaciona un pixel con el superpixel al que pertenece.

En el siguiente diagrama, se detalla el método de cálculo de las imágenes reducidas mediante superpixels. Es importante observar que se generan miniaturas tanto para la imagen original, como para su máscara.



Las imágenes resultantes se pueden entonces utilizar por el resto de algoritmos sin ningún tipo de problemas

3.2.2 Random Walks

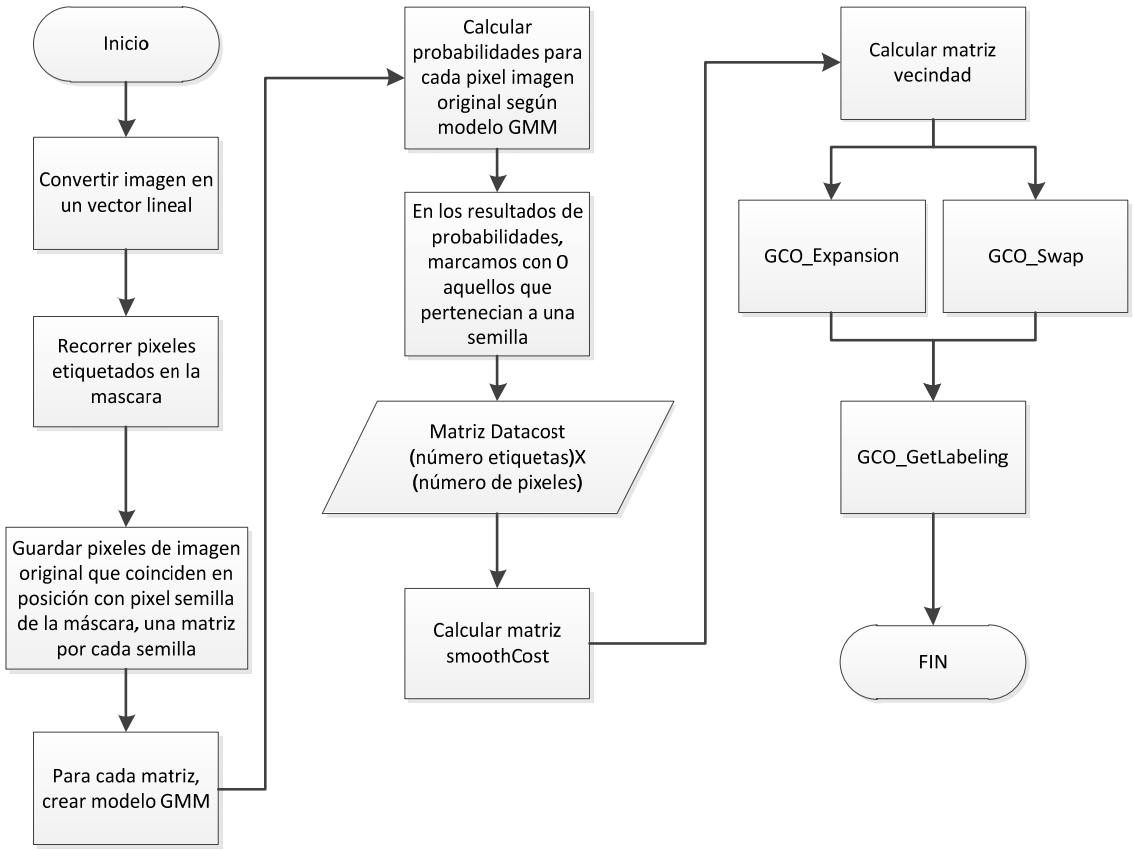
La implementación de Random Walks (10) utilizada soluciona el tiempo de cálculo que necesita el algoritmo original. Se han realizado pequeñas modificaciones en el código para adecuarlo a los parámetros utilizados, es decir, la imagen y su máscara correspondiente. Gracias a esta adaptación, retorna una matriz MxN de igual tamaño que la imagen original, con el etiquetado de cada uno de los pixeles.

3.2.3 α-β swap y α-Expansion

Para poder implementar estos dos algoritmos, se ha utilizado la librería gco-v3.0 realizada por Olga Veksler y Andrew Delong (11).

Esta librería se basa en una parte codificada en forma de librerías en c, que una vez compilada generan unos archivos dll utilizables desde Matlab. La otra parte es un "envoltorio" o como se llama en inglés, wrapper. Son una serie de archivos que se encargan de recibir las llamadas realizadas en código Matlab y traspasarlas a los archivos dll, y viceversa.

Al utilizar los dos algoritmos la misma librería y compartir procedimientos, el procedimiento es el mismo, únicamente hay que cambiar una línea de código, para diferenciar si se llama a α-β swap o a α-Expansion. El siguiente diagrama muestra el funcionamiento de los dos algoritmos.



La explicación del funcionamiento es la siguiente:

1. El primer paso es adaptar la imagen y convertirla a una matriz 1x(número de pixeles) ya que ese es el formato que utiliza la librería gco-v3.0
2. Se recorre la máscara de entrada, y para cada pixel que pertenece a una etiqueta, se almacena el pixel de la imagen original. Este se guarda en una matriz diferente para cada etiqueta. De esta manera, se tienen todos los pixeles de cada etiqueta agrupados en matrices diferentes.
3. Para cada una de estas matrices, se crea un Gaussian Mixture Model (GMM). Este es un modelo probabilístico que utilizará los valores RGB delos pixeles seleccionados como información para modelar las diferentes clases.
4. El objeto GMM (Gaussian Mixture Model) de Matlab dispone de una función llamada pdf, la cual, al ser utilizada con la matriz del paso número 1 nos devolverá la probabilidad de cada uno de los pixeles de pertenecer a esa etiqueta.
5. Juntando las diferentes matrices de probabilidad, se obtiene una matriz de tamaño (número de etiquetas)X(número de pixeles)

6. 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 4 etiquetas será esta:

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

7. La matriz de vecindad es una matriz de tamaño (número de pixeles) X (número de pixeles) donde se indica el valor de vecindad entre un pixel y los que le rodean. Normalmente, en esta matriz, para cada fila (pixel) solo habrá 8 valores (cada pixel tiene 8 vecinos) y el resto se quedarán con el valor a cero. La función de vecindad, calcula el peso de las aristas entre 2 píxeles vecinos, llamados "m" y "n", y se basa en la diferencia de color entre estos 2 píxeles como muestra la siguiente ecuación

$$V(\underline{\alpha},\mathbf{z}) = \gamma \sum_{(m,n)\in\mathbf{C}} dis(m,n)^{-1}\,[\alpha_n \neq \alpha_m] \exp -\beta(z_m - z_n)^2$$

8. Todos estos valores son utilizados para, al realizar la llamada a α-β swap o a α-Expansion, obtener el etiquetado de la imagen.

Debido a que la matriz de vecindad puede contener muchos valores, a veces se producen errores por falta de memoria. Se ha fijado que estos dos métodos serán llamados siempre con la pre-segmentación en superpixels, para poder trabajar con imágenes más pequeñas.

3.3 Métricas utilizadas

Para poder evaluar el funcionamiento de los diversos algoritmos, se ha definido un método para comparar los resultados obtenidos y las imágenes pre-etiquetadas de la librería Human Limb (1).

Tal y como se explica en el punto 3.1, se ha definido una tabla que relaciona las 4 etiquetas que generarán los algoritmos respecto a las etiquetas ya marcadas en la librería Human Limb.

Para evaluar la eficiencia se genera una matriz de confusión para evaluar cada una de las etiquetas. Esta matriz se rellena indicando para cada etiqueta de la imagen de referencia, cuantos píxeles han sido etiquetados como tal. Un resultado óptimo es aquel donde los mayores valores se concentren en la diagonal de la matriz.

Tabla 3-2

	Label 1	Label 2	Label 3	Label 4
Label 1				
Label 2				
Label 3				
Label 4				

Para poder calcular un porcentaje de eficiencia que resuma en un solo valor cómo de bueno es el método, lo que se hace es sumar los valores de la diagonal y dividir por la suma total de la matriz.

3.4 Resultados obtenidos

Una vez definidos los métodos y métricas que se van a utilizar, se ha lanzado el bucle principal con 10 imágenes de la librería Human Limb, y cada una de estas imágenes a su vez tiene asignadas 10 máscaras. El proceso ha sido adaptado para generar un archivo csv con todos los datos capturados de las variables intermedias.

Este archivo está separado en columnas, por el símbolo del punto y coma. Un ejemplo de línea es el siguiente.

Tabla 3-3

Imagen	p005_005.bmp
mascara	m_1_p005_005.bmp
num_mascara	1
Tipo mascara	normal
Modo	RW
l1-1	40690
l1-2	597
l1-3	6608
l1-4	1234
l2-1	11172
l2-2	3420
l2-3	5872
l2-4	0
l3-1	245
l3-2	0
l3-3	10686
l3-4	43
l4-1	13977
l4-2	566
l4-3	109
l4-4	21872
porcentaje	65,48

- Imagen: archivo con la imagen a segmentar
- Mascara: nombre del archivo de máscara utilizado
- Número de máscara: En este caso iba del 1 al 10, sirve para ordenar los resultados
- Tipo máscara: nos informa de si la máscara es normal, o ha sido combinada con las anteriores.
- Modo: Nos dice si la línea ha sido de Random Walks, Expansion o Swap.
- L1-1 a L4-4: celdas de la tabla de confusión con los resultados.
- Porcentaje: Valor de eficiencia del resultado respecto la imagen etiquetada manualmente.

Aparte de almacenar todos los valores en un archivo para poder generar las estadísticas, también se guardan imágenes de los resultados para evaluar gráficamente como ha sido cada etiquetado.

Estas imágenes son un duplicado de la original, donde se ha superpuesto en forma de colores semitransparentes los resultados obtenidos, y se marcan además, las semillas de las máscaras utilizadas, que se pueden ver como pequeños cuadrados en las imágenes adjuntas.

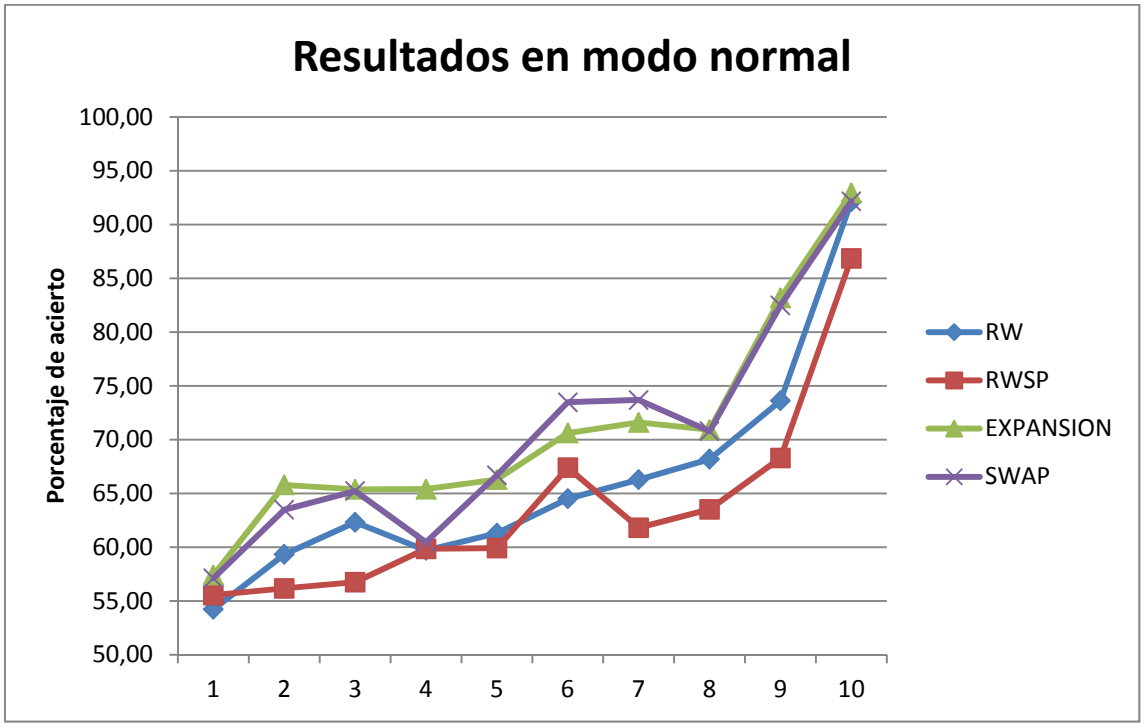
Como se ve en la siguiente pareja de imágenes, un mayor uso de etiquetas conlleva por norma general unos resultados mejores.

Resultado RW con pocas semillas [Imagen 3-4]

Resultado RW con alta densidad de semillas [Imagen 3-5]

3.4.1 Resultados cuantitativos

A partir de todos los datos exportados al archivo en formato csv, se ha generado los siguientes gráficos donde se visualiza los resultados obtenidos. Se muestran la media de los porcentajes de eficiencia por cada número de máscara:



Máscaras	RW	RWSP	EXPANSION	SWAP	Total general
1	54,22	55,54	**57,40**	57,11	**56,07**
2	59,33	56,17	**65,79**	63,49	**61,19**
3	62,32	56,75	**65,38**	65,22	**62,42**
4	59,71	59,85	**65,40**	60,46	**61,35**
5	61,30	59,91	66,29	**66,69**	**63,55**
6	64,52	67,40	70,61	**73,48**	**69,00**
7	66,29	61,81	71,61	**73,69**	**68,35**
8	68,19	63,52	**70,92**	70,79	**68,36**
9	73,62	68,30	**83,18**	82,48	**76,90**
10	92,12	86,86	**92,95**	92,17	**91,03**
Total general	66,16	63,61	**70,95**	70,56	**67,82**

Resultados de aplicar las máscaras de forma normal. El eje de las X informa del número de la máscara [Gráfica 1]

En la gráfica 1 se puede observar como los resultados de los algoritmos α-β Swap y α-Expansion tienen una mayor eficiencia en comparación con Random Walks. Entre ellos dos, no hay grandes diferencias, y es de suponer que esa alternancia por ser el más efectivo para cada máscara irá en función del tipo o distribución de las semillas por la imagen.

También se observa que al ejecutar Random Walks si se pre-segmenta la imagen mediante superpixels provoca una reducción de la eficiencia de este algoritmo.

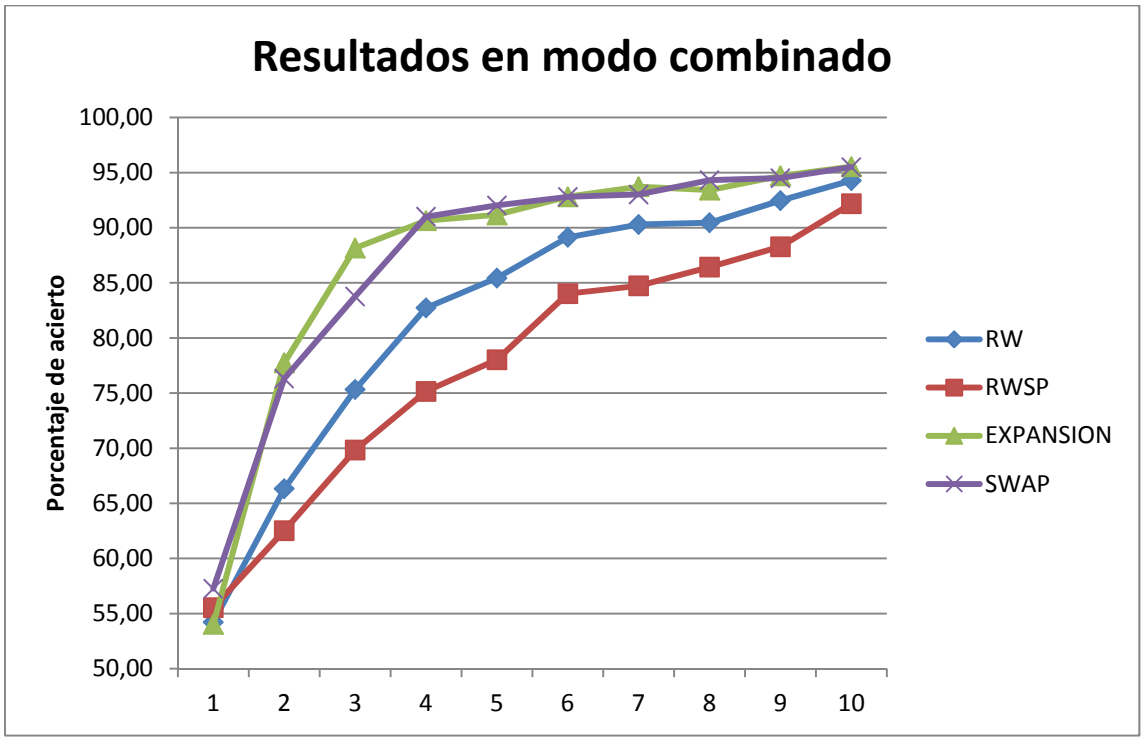
La subida de la eficiencia de todos los algoritmos para la máscara 9 y 10 se debe a que estas dos máscaras contienen una mayor densidad de semillas, ya que se ha querido analizar también el funcionamiento de los métodos cuando la interacción ha sido alta.

La gráfica 2 muestra los mismos cálculos que la gráfica 1, pero en este caso se muestran los resultados para el caso de la utilización de las máscaras combinadas.

Tal y como se puede observar, los diferentes algoritmos aumentan bastante su eficiencia gracias a una mayor interacción. Siguen generando mejores resultados aquellos algoritmos basados en graphcut, pero ahora las diferencias entre α-β Swap y α-Expansion se han reducido y muestran una eficiencia muy similar.

El comportamiento de Random Walks sigue siendo superior cuando se utiliza sin pre-segmentar la imagen.

La progresión de la eficiencia en función del grado de interacción es diferente según el algoritmo. Los que se basan en graphcut acusan una gran mejora, llegando a un 90% de eficiencia para la 4 iteración. En cambio, Random Walks mejora continuamente, pero no llega a un 90% de eficiencia hasta la 7 iteración, y si se utiliza la pre-segmentación necesita de 10 iteraciones para poder igualar ese valor.



Máscaras	RW	RWSP	EXPANSION	SWAP	Total general
1	54,22	55,54	54,03	**57,26**	**55,26**
2	66,32	62,53	**77,75**	76,32	**70,73**
3	75,32	69,84	**88,16**	83,77	**79,27**
4	82,73	75,16	90,64	**91,00**	**84,89**
5	85,44	78,02	91,18	**92,03**	**86,67**
6	89,13	84,02	**92,81**	92,80	**89,69**
7	90,29	84,74	**93,72**	93,02	**90,44**
8	90,46	86,43	93,42	**94,31**	**91,15**
9	92,46	88,28	**94,71**	94,49	**92,49**
10	94,28	92,19	**95,54**	95,50	**94,38**
Total general	82,07	77,68	**87,20**	87,05	**83,50**

Resultados de aplicar las máscaras de forma combinada. El eje de las X informa del número de la máscara [Gráfica 2]

Tal y como se comenta en la sección 3.3 Métricas utilizadas, se ha generado una matriz para cada segmentación realizada. Todas estas matrices se han resumido en las dos siguientes tablas. Estas tabas muestran los valores en forma de porcentaje para cada etiqueta de ground-truth. El color de la celda va del blanco al verde en función del valor de esta. Para una eficiencia del 100% solo está en color verde la diagonal de la matriz.

En la siguiente tabla se muestra el resumen en función del tipo de máscara, comparando los resultados sin tener en cuenta el algoritmo utilizado. Se observa cómo el etiquetado de la etiqueta cabeza es el peor resultado de los dos modos, aunque en modo normal incluso el porcentaje de error supera al etiquetaje correcto. También resulta extraño el hecho que en modo combinado, el porcentaje de etiquetas erróneas Cabeza-Torso (14,69) resulta superior al correspondiente en modo normal (12,07).

Resultado Algoritmos (en porcentaje)						
		Fondo	Cabeza	Torso	Piernas	
Ground-truth	Fondo	93,60	1,32	1,95	3,13	Modo Normal
	Cabeza	44,98	39,31	12,07	3,64	
	Torso	16,76	1,57	77,36	4,31	
	Piernas	19,53	0,70	3,65	76,13	
	Fondo	96,15	0,72	1,04	2,10	Modo Combinado
	Cabeza	30,32	53,04	14,69	1,95	
	Torso	10,90	1,00	86,04	2,07	
	Piernas	11,51	0,51	0,66	87,32	

Tabla 3-4

La tabla que sigue a continuación muestra los porcentajes de las matrices de confusión agrupadas por el algoritmo de segmentación. Aunque en este caso no se evalúa el modo de utilización de la máscara, se observan las dos circunstancias anteriores. Es decir, para los algoritmos de Random Walks, el etiquetado de cabeza produce errores etiquetando como

Fondo, y para los algoritmos de α-EXPANSION y α-β SWAP, aunque tienen una mejor eficiencia, los valores de las etiquetas erróneas Cabeza-Torso son superiores a las correspondientes para los algoritmos Random Walks.

Resultado Algoritmos (en porcentaje)						
		Fondo	Cabeza	Torso	Piernas	
Ground-truth	Fondo	96,56	0,59	0,93	1,93	α-EXPANSION
	Cabeza	29,41	51,33	15,50	3,76	
	Torso	10,16	0,96	85,25	3,63	
	Piernas	6,18	0,15	3,05	90,62	
	Fondo	93,96	1,66	1,74	2,64	RW
	Cabeza	43,09	44,05	11,28	1,58	
	Torso	18,28	1,88	76,93	2,91	
	Piernas	20,58	0,74	0,67	78,02	
	Fondo	92,65	1,21	2,31	3,83	RWSP
	Cabeza	48,33	37,72	11,64	2,31	
	Torso	17,12	1,41	78,94	2,54	
	Piernas	28,93	1,35	1,81	67,92	
	Fondo	96,32	0,62	1,00	2,06	α-β SWAP
	Cabeza	29,77	51,60	15,11	3,52	
	Torso	9,76	0,88	85,68	3,68	
	Piernas	6,38	0,18	3,09	90,35	

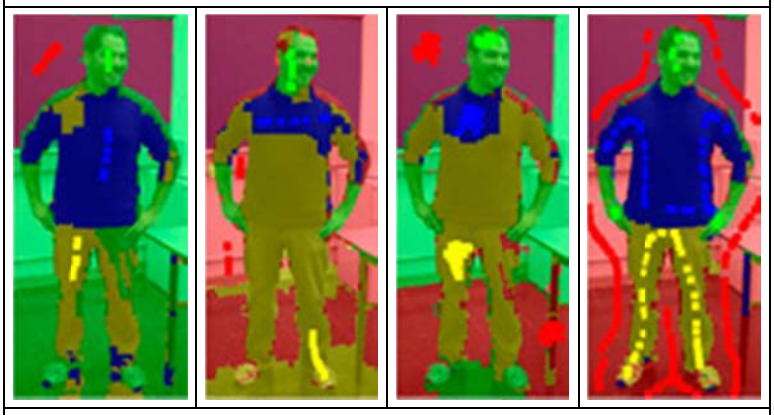
Tabla 3-5

3.4.2 Resultados cualitativos

A continuación, se muestra el resultado de las diferentes segmentaciones para una de las 10 imágenes, de esta forma se puede observar cual ha sido el resultado obtenido en la imagen. Dado que el número de figuras resulta elevado, en la tabla figura la máscara 1,2, 5 y 10 para cada método, tanto cuando el uso de las máscaras es el normal o es en el modo combinado

Resultados gráficos para máscaras normales			
1	**2**	**5**	**10**
			
Random Walks			

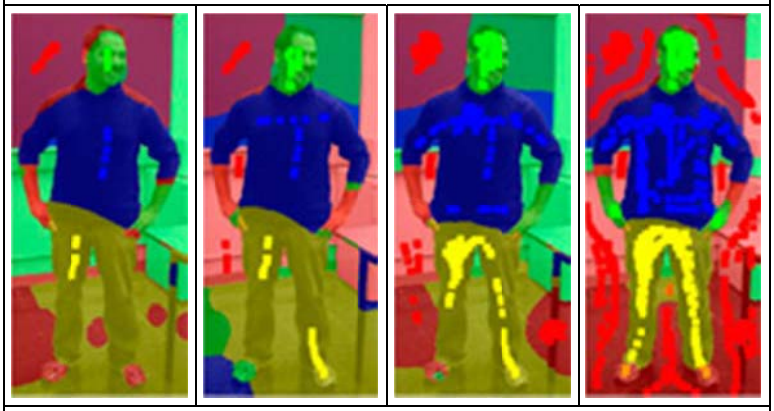
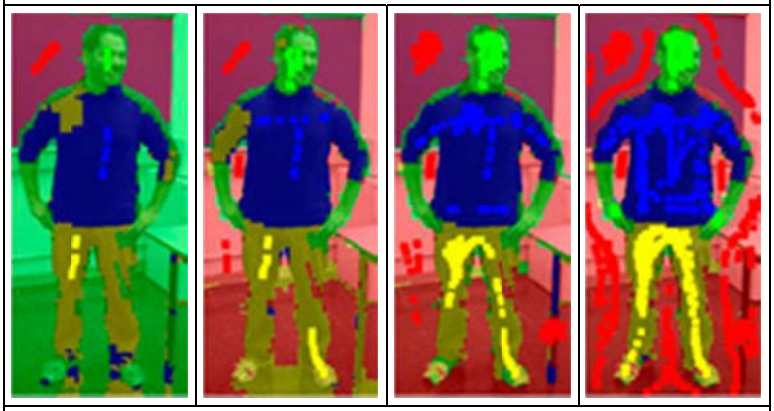
Random Walks utilizando Superpixels

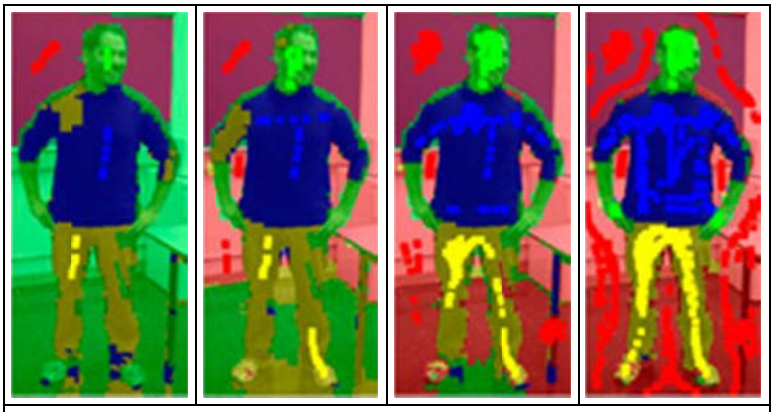


α-Expansion

α-β Swap

Comparativa de las diferentes segmentaciones normales aplicadas a una imagen en particular [Tabla 3-6]

Resultados Gráficos para máscaras combinadas			
1	**2**	**5**	**10**
			
Random Walks			
			
Random Walks utilizando Superpixels			
			
α-Expansion			



α-β Swap

Comparativa de las diferentes segmentaciones combinadas aplicadas a una imagen en particular [Tabla 3-7]

En los resultados de la máscara 5 de α-β Swap y α-Expansion, se puede observar claramente la diferencia de utilizar las máscaras de forma normal o combinada.

4 Conclusiones

En este proyecto, se ha desarrollado un software para la comparación de diferentes métodos de segmentación aplicados concretamente a la segmentación de personas. A partir de los experimentos realizados he sacado diferentes conclusiones, relacionadas con diferentes aspectos del desarrollo del proyecto.

En primer lugar, se ha observado cómo la interacción del usuario influye de manera substancial en el resultado de los algoritmos de segmentación

También ha quedado reflejado que para este tipo de imágenes, los algoritmos basados en graphcut (α-Expansion y α-β Swap) generan un resultado mejor, que no el algoritmo de Random Walks. Esto parece deberse a que los algoritmos α-Expansion y α-β Swap confian bastante en el color de la imagen e introducen coherencia espacial en el etiquetado, teniendo en cuenta las etiquetas de los píxeles vecinos, así como la diferencia de apariencia (en este caso, color) entre dichos píxeles. Por otro lado, Random Walks tiende a tener más en cuenta la proximidad a las semillas.

Esta diferencia en la eficiencia a favor de los algoritmos basados en graphcut se ve aumentada cuando se utilizan las máscaras de forma combinada, ya que mientras para el caso de α-Expansion y α-β Swap con cuatro interacciones se sobrepasa el 90% de eficiencia, el método Random Walks no llega a este valor hasta tener diez interacciones. También hay que decir, que la aplicación de superpixels, al menos para Random Walks no conlleva una mejora de efectividad, ya que los resultados siempre son inferiores a los ejemplos que no lo utilizan.

En resumen, una correcta aplicación de estos algoritmos, sobre todo los α-Expansion y α-β Swap puede facilitar bastante la detección de personas, pudiendose aplicar en conjunto con algoritmos de detección de personas (que no de segmentación).

En este proyecto , se ha desenvolupado un software de forma que hay una única función o método encargada de llamar a los diferentes algoritmos en función de los parámetros que se pasen, unificando y simplificando el desarrollo. Este trabajo podrá ser continuado si así interesase, realizando pequeñas modificaciones, para adaptarlo a los diferentes entornos que pudiese haber. . Como trabajo futuro, se puede ampliar el estudio estudiando el caso de que las máscaras puedan contener errores, etiquetando zonas incorrectamente. Una posible solución a esto podría realizarse teniendo en cuenta el número de veces que se ha etiquetado el mismo punto, para eliminar aquellas semillas que se han marcado mal una vez.

A título personal, gracias a la realización de este proyecto, he ampliado mis conocimientos sobre nuevos campos de aplicación de la informática, ya que conocia cosas sobre visión artificial, pero nunca habia tenido ocasión de ponerlas en práctica.

En general, el desarrollo de este proyecto ha sido ameno, y el uso de la herramienta Matlab ha facilitado en gran manera su finalización. La gran mayoria de métodos se han encontrado codificados en este lenguaje, y su utilización y aprendizaje ha resultado más comodo que si se hubiese realizado en otro lenguaje, como el c++ utilizando OpenCV. Si es cierto que su velocidad es mucho menor, aunque este no era un punto crítico del proyecto.

5 Bibliografía

1. *Human Limb DataSet*. http://www.maia.ub.es/~sergio/linked/humanlimbdb.zip.

2. Mori, Greg. http://www.cs.sfu.ca/~mori/research/superpixels.

3. Moore, Alastair; J D Prince, Simon; Warrell, Jonathan; Mohammed, Umar; Jones, Graham. *Superpixel Lattices*. 2008. CVPR. http://www.cs.ucl.ac.uk/staff/s.prince/Papers/SuperpixelLattices.pdf.

4. Comaniciu, Dorin y Meer, Peter. *Mean shift: A robust approach toward feature space analysis*. In PAMI. 2002, págs. 603--619.

5. *Mean Shift Segmentation in Matlab*. http://www.shawnlankton.com/2007/11/mean-shift-segmentation-in-matlab/.

6. Cour, Timothee, Yu, Stella y Shi, Jianbo. *Normalized Cuts Segmentation Code, for MATLAB*. University of Pennsylvania, Computer and Information Science Department, 2004. http://www.seas.upenn.edu/~timothee/software/ncut/ncut.html.

7. Rother, Carsten, Kolmogorov, Vladimir y Blake, Andrew. *GrabCut: Interactive Foreground Extraction using Iterated Graph Cuts*. 2004, ACM Transactions on Graphics, Vol. 23, págs. 309--314.

8. *OpenCV.* http://opencv.willowgarage.com/wiki/.

9. Boykov, Yuri, Veksler, Olga y Zabih, Ramin .*Efficient Approximate Energy Minimization via Graph Cuts*. 12, Nov de 2001, IEEE TPAMI, Vol. 20, págs. 1222-1239.

10. Andrews, Shawn, Hamarneh, Ghassan y Saad, Ahmed . *Fast Random Walker with Priors using Precomputation for Interactive Medical Image Segmentation*. Lecture Notes in Computer Science, Medical Image Computing and Computer-Assisted Intervention (MICCAI). 2010. {http://fastrw.cs.sfu.ca/miccai2010b.pdf.

11. Veksler, Olga y Delong, Andrew. *Multi-label optimization*. 28 de 4 de 2010. http://vision.csd.uwo.ca/code/gco-v3.0.zip.

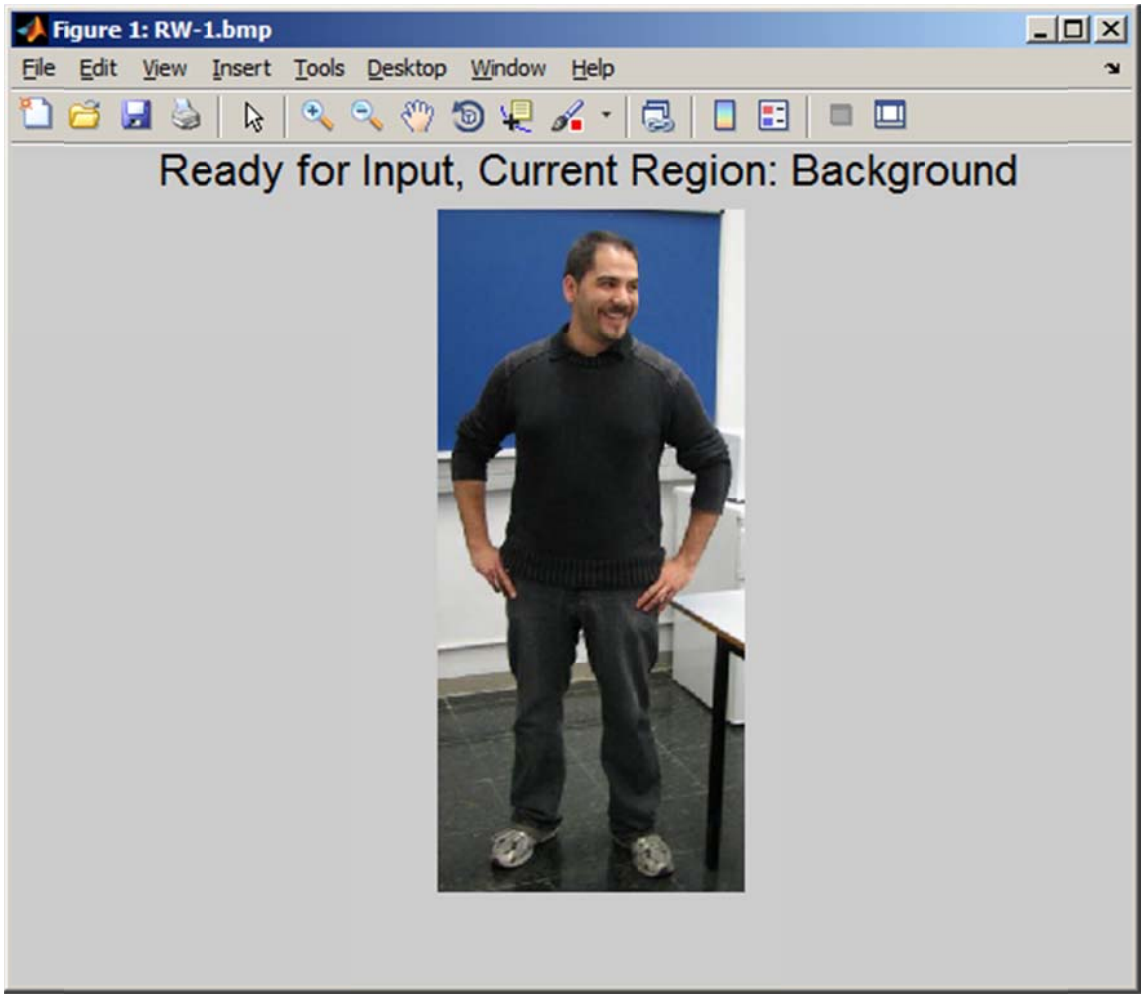
12. Andrews, Shawn, Hamarneh, Ghassan y Saad, Ahmed. *Fast Random Walker with Priors using Precomputation for Interactive Medical Image Segmentation*. [ed.] Simon Fraser University, Burnaby, BC, Canada School of Computing Science. June de 2010. TR 2010-07, págs. 1-13. http://www.cs.sfu.ca/~hamarneh/ecopy/sfu_cs2010_07.pdf.

6 Anexos

6.1 GeneraMasks

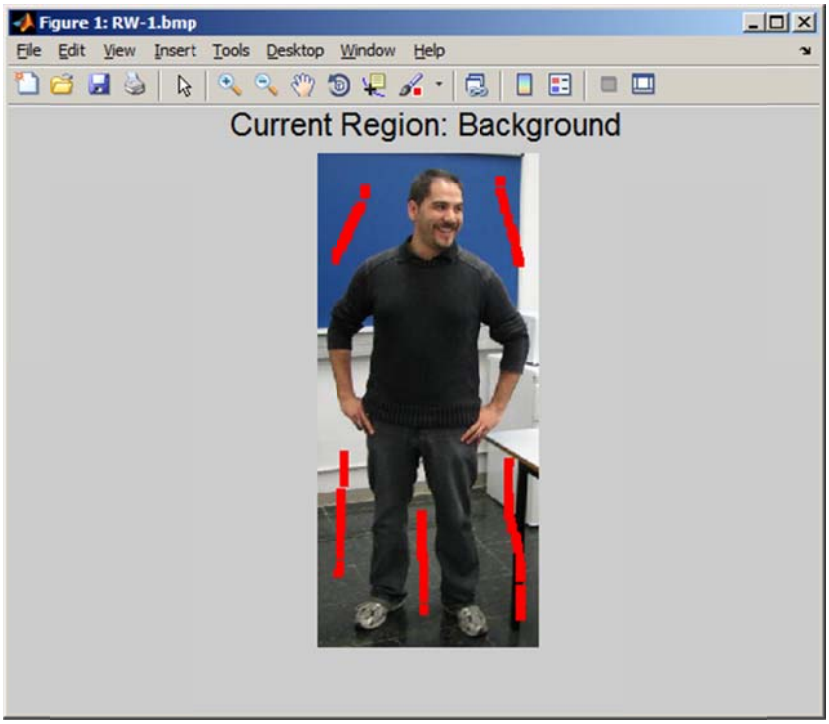
Dado el gran número de máscaras que se han generado, se ha codificado una función en Matlab encargada de realizar este proceso ayudando al usuario en esta función. Se ha adaptado parte del código de Random Walks que permitía crear las semillas directamente en la imagen, de tal forma que se pueden definir las cuatro semillas definidas, y se almacenan en la carpeta correspondiente.

El funcionamiento es sencillo, se llama a esta función con el nombre de la imagen y a continuación se abre una ventana con la imagen.



Generador de máscaras esperando para etiquetar el fondo [Imagen 6-1]

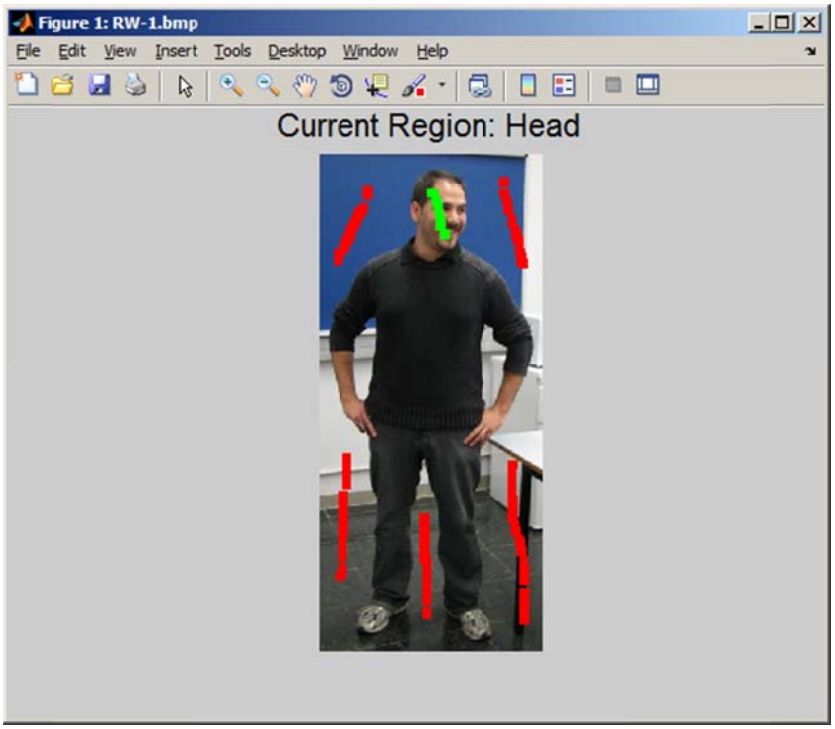
En la pantalla se nos informa que se va a etiquetar el fondo (background), por lo que hay que ir marcando con el ratón aquellas zonas de fondo que nos interesen. Dado que se han creado las funciones de cero, el grosor de las etiquetas se realizan con dos bucles anidados que van de x-10 a x+10 y Y-10 a Y+10, por lo que la velocidad no es del todo optima, y no hay que hacer movimientos rápidos del ratón.



Generador de máscaras donde ya se ha seleccionado el fondo [Imagen 6-2]

En esta imagen se puede observar una vez se ha marcado con el ratón las semillas del fondo.

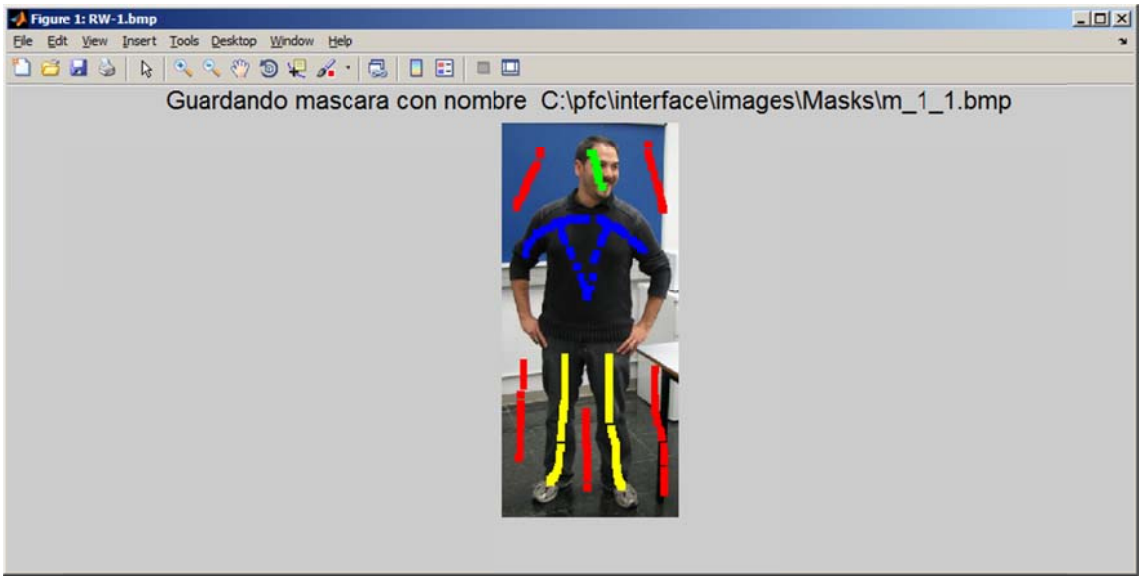
A continuación hay que pulsar una sola vez el botón derecho, y se cambiará la región actual a la cabeza.



Generador de máscaras donde ya se ha etiquetado la cabeza [Imagen 6-3]

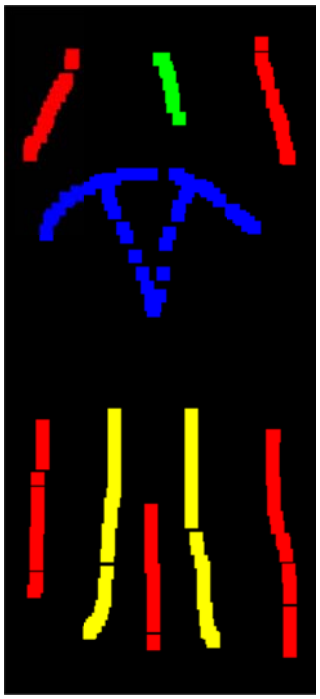
De esta forma, al volver a pulsar el botón derecho, se cambia a la región del torso y luego a los pies. Una vez se han marcado todas las semillas, se debe hacer doble clic con el botón derecho,

y la función nos almacena la máscara en la carpeta correspondiente, lista para utilizar. Una vez se han realizado 10 máscaras, el sistema se detiene.



Generador de máscaras, cuando ha guardado una máscara con el nombre m_1_1.bmp [Imagen 6-4]

La máscara resultante queda de la siguiente forma:



Máscara generada [Imagen 6-5]

6.2 Contenido del CD-ROM

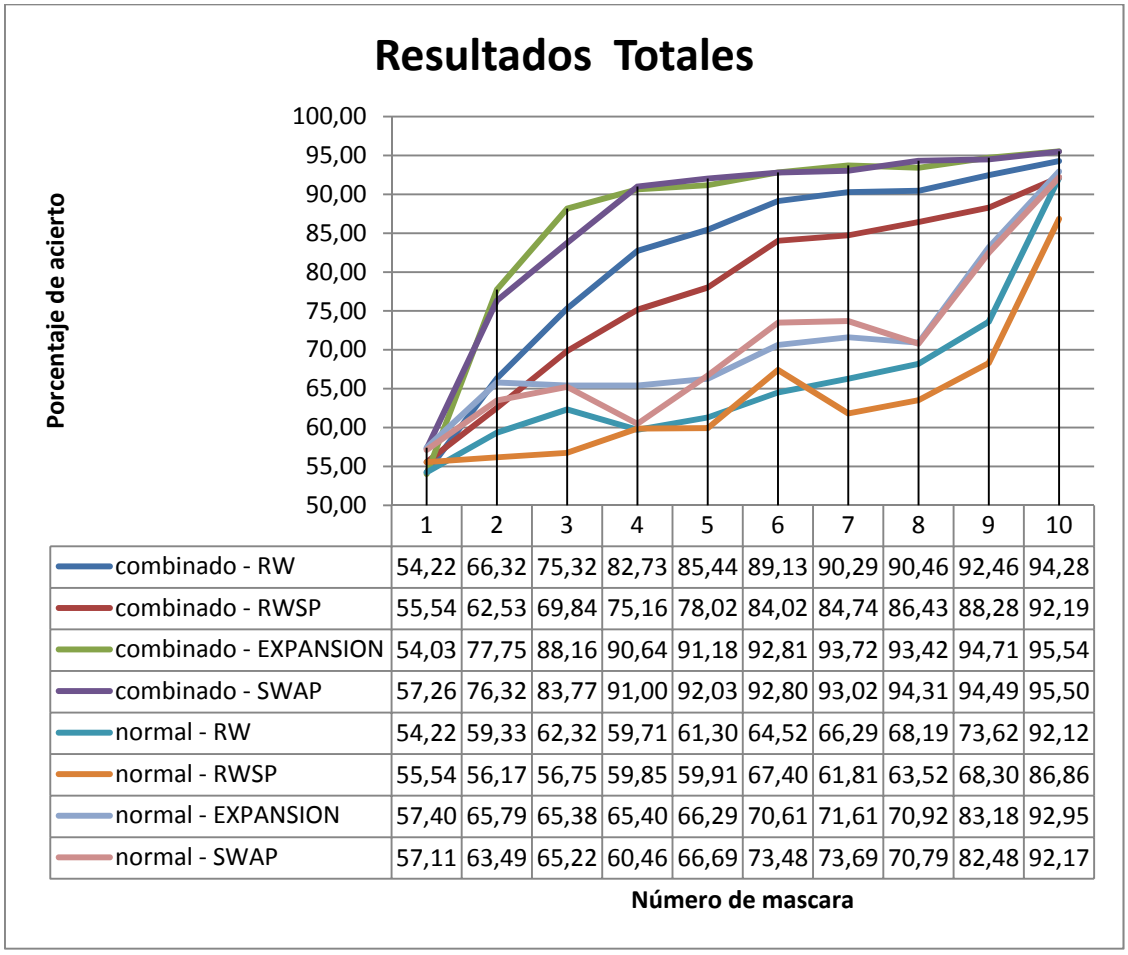
El contenido del CD-ROM es el siguiente:

- gco-v3.0 --> librerías de matlab para poder realizar la segmentación A-B Swap y A-Expansion
- Grabcut --> código fuente modificado del ejemplo grabcut incluido con Opencv, para que permita la entra de máscaras mediante archivo
- graphAnalysisToolbox-1.0 --> Librería de herramientas utilizada por el algoritmo Random Walks
- HumanLimbDB --> Base de datos de imágenes adjuntando la segmentación previa para evaluar los resultados.
- Interface --> Carpeta que incluye los archivos codificados para realizar todas las pruebas
    - Bucle.m: archivo encargado de llamar al archivo SI.m por cada imagen, modo y mascara, aparte de generar las máscaras combinadas.
    - Gco.m: código fuente de la segmentación α-Expansion.
    - gcoSwap.m: código fuente de la segmentación α-β Swap.
    - Grabc.exe: ejecutable del algoritmo binario grabcut.
    - grlC.m y grl.mexw32: código matlab y librería compilada que generan los superpixels.
    - Recorta4.m, se encarga de generar la imagen segmentada en resolución normal a partir de la segmentación de una imagen en superpixels.
    - Resultados.csv: archivo que contiene los valores de los resultados obtenidos por los diferentes algoritmos.
    - SI.m: archivo de interfaz que aglutina todas las llamadas a los diferentes algoritmos, en función de los parámetros que se le pasen, generando imágenes de resultados y exportando valores a un archivo csv.
    - FastRW: Carpeta con el código del algoritmo Random Walks.
    - NCutImage: carpeta con el código del algoritmo ncuts, no utilizado para el análisis.
    - Rw: Carpeta con el código de una implementación alternativa a Random Walks, pero tampoco ha sido utilizada en el análisis.
    - Images: Carpeta donde se encuentran todas las imágenes, tanto las de personas, máscaras, imágenes pre-segmentadas de la librería Human Limb e imágenes de resultados de la segmentación
        - GeneraMask.m: Archivo encargado de facilitar la tarea de generar las máscaras.
- Memoria.docx --> Esta memoria en formato Word

-  Resultados totales.xlsx --> Archivo Excel que incluye todos los datos de los resultados generados.
-  Segmentación interactiva multi-clase de personas.pptx: Presentación del proyecto.

6.3 Listados de imágenes y gráficos

La siguiente gráfica muestra todos los resultados obtenidos en una única gráfica, para así poder comparar mejor los resultados.

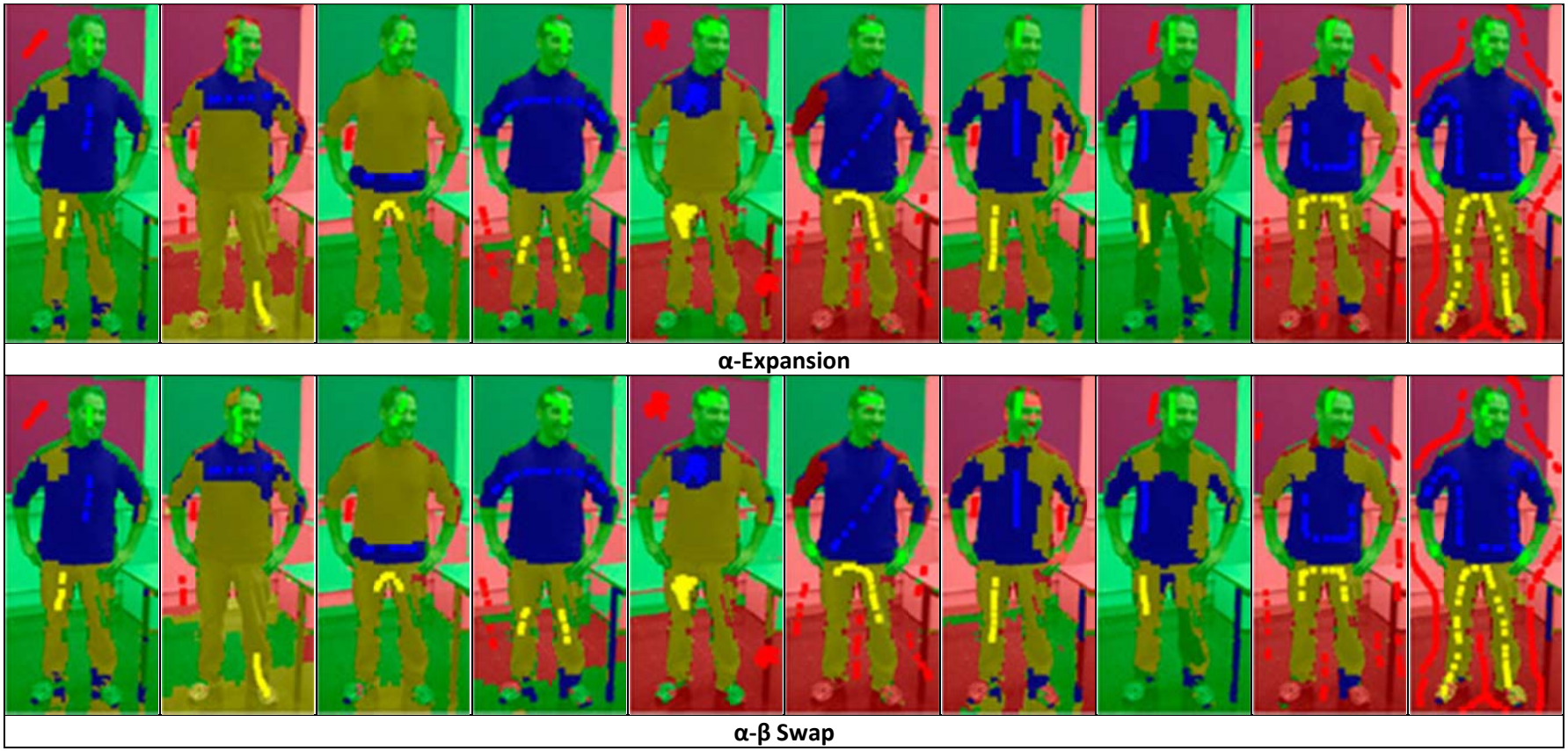


	1	2	3	4	5	6	7	8	9	10
combinado - RW	54,22	66,32	75,32	82,73	85,44	89,13	90,29	90,46	92,46	94,28
combinado - RWSP	55,54	62,53	69,84	75,16	78,02	84,02	84,74	86,43	88,28	92,19
combinado - EXPANSION	54,03	77,75	88,16	90,64	91,18	92,81	93,72	93,42	94,71	95,54
combinado - SWAP	57,26	76,32	83,77	91,00	92,03	92,80	93,02	94,31	94,49	95,50
normal - RW	54,22	59,33	62,32	59,71	61,30	64,52	66,29	68,19	73,62	92,12
normal - RWSP	55,54	56,17	56,75	59,85	59,91	67,40	61,81	63,52	68,30	86,86
normal - EXPANSION	57,40	65,79	65,38	65,40	66,29	70,61	71,61	70,92	83,18	92,95
normal - SWAP	57,11	63,49	65,22	60,46	66,69	73,48	73,69	70,79	82,48	92,17

Resultados de los 2 modos agrupados [Gráfica 3]

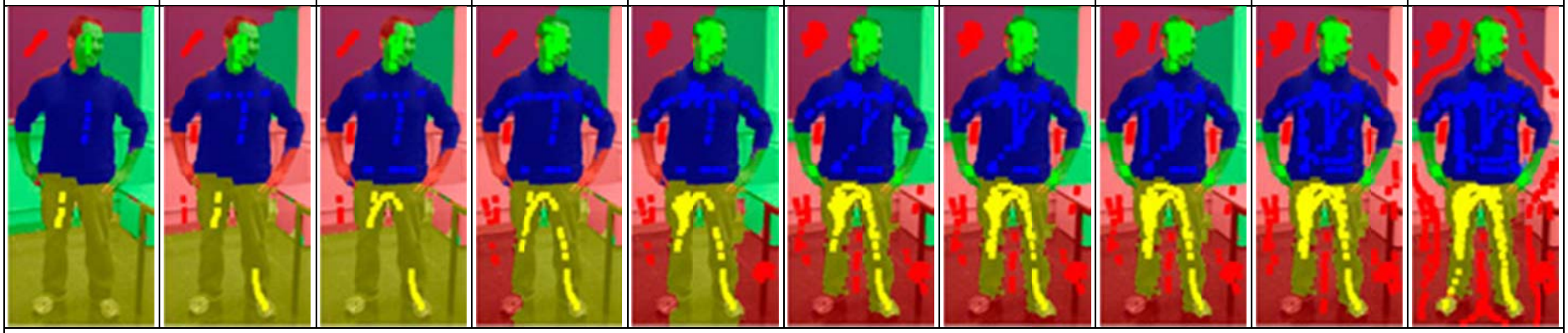
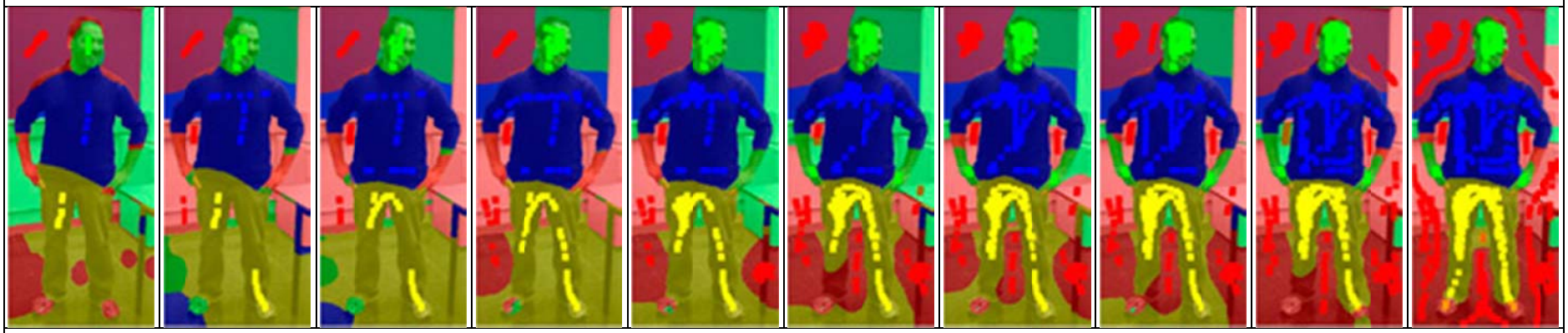
Estos son los resultados del punto 3.4.2Resultados cualitativos para todas las etiquetas utilizadas:

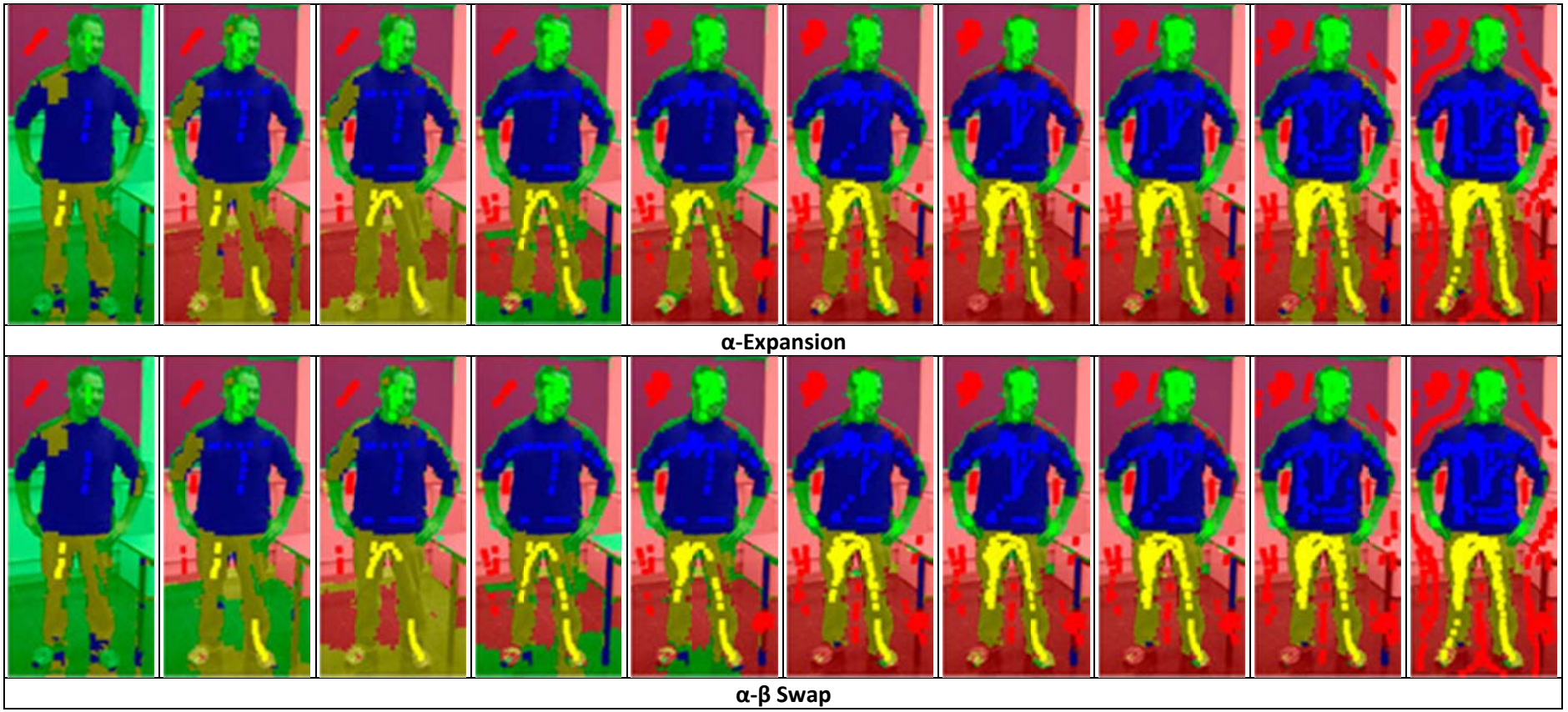
Resultados gráficos para máscaras normales									
1	2	3	4	5	6	7	8	9	10
									
Random Walks									
									
Random Walks utilizando Superpixels									



α-Expansion
α-β Swap

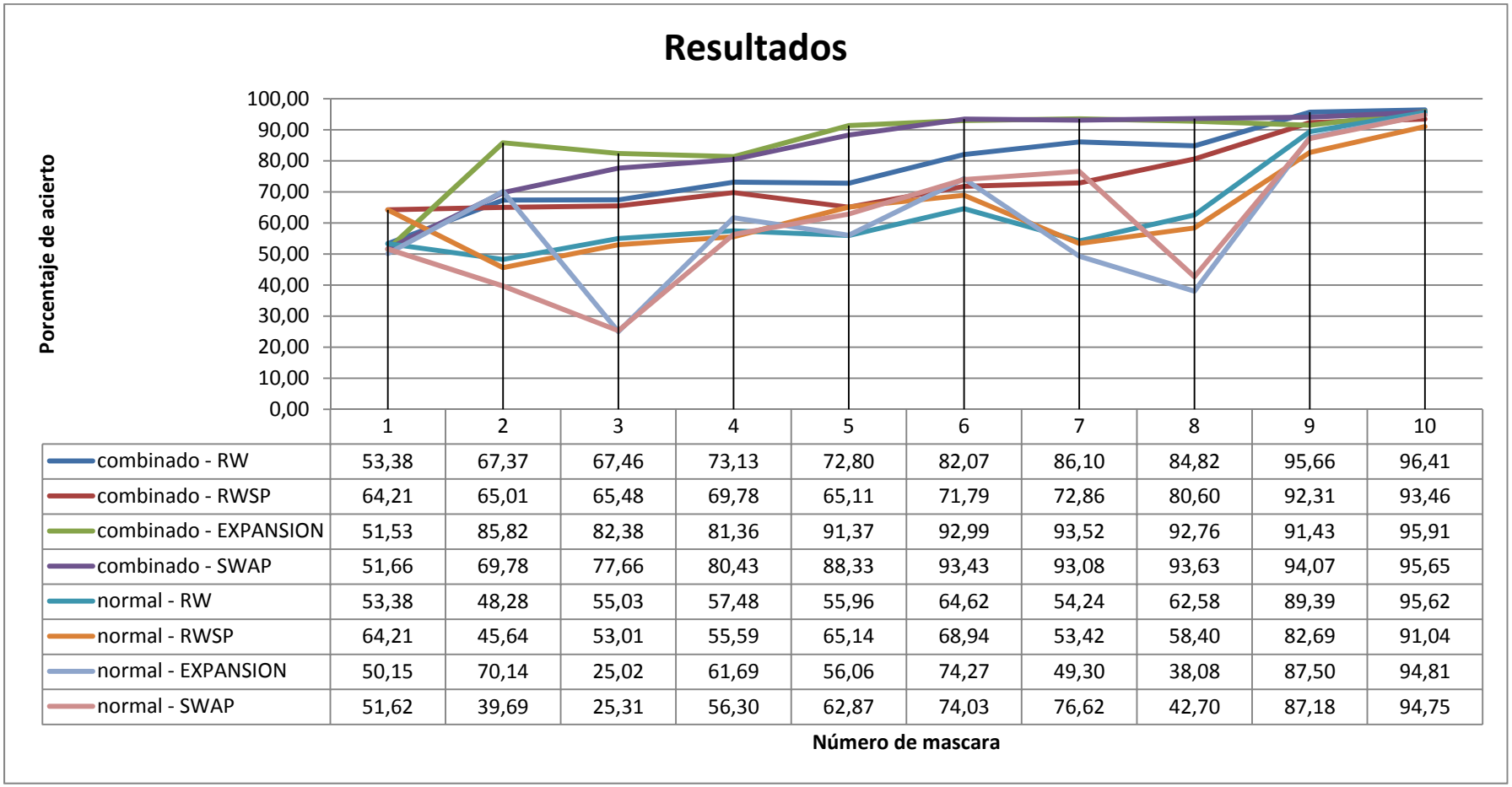
Comparativa de las diferentes segmentaciones normales aplicadas a una imagen en particular [Tabla 6-1]

Resultados Gráficos para máscaras combinadas									
1	2	3	4	5	6	7	8	9	10
									
Random Walks									
									
Random Walks utilizando Superpixels									



Comparativa de las diferentes segmentaciones combinadas aplicadas a una imagen en particular [Tabla 6-2]

Los resultados para esta imagen en particular se pueden visualizar en la siguiente gráfica:



	1	2	3	4	5	6	7	8	9	10
combinado - RW	53,38	67,37	67,46	73,13	72,80	82,07	86,10	84,82	95,66	96,41
combinado - RWSP	64,21	65,01	65,48	69,78	65,11	71,79	72,86	80,60	92,31	93,46
combinado - EXPANSION	51,53	85,82	82,38	81,36	91,37	92,99	93,52	92,76	91,43	95,91
combinado - SWAP	51,66	69,78	77,66	80,43	88,33	93,43	93,08	93,63	94,07	95,65
normal - RW	53,38	48,28	55,03	57,48	55,96	64,62	54,24	62,58	89,39	95,62
normal - RWSP	64,21	45,64	53,01	55,59	65,14	68,94	53,42	58,40	82,69	91,04
normal - EXPANSION	50,15	70,14	25,02	61,69	56,06	74,27	49,30	38,08	87,50	94,81
normal - SWAP	51,62	39,69	25,31	56,30	62,87	74,03	76,62	42,70	87,18	94,75

Resultados de segmentación de una imagen en particular [Gráfica 4]