Proyecto de grado
"Análisis de video en Biomecánica"

Gonzalo Pereira. Mauricio Ramos, Guillermo Ottado, Andréi Guchin.

Índice

1. Introducción

1.1. Estado del arte

2. Implementación de bloques del sistema

3. Segmentación

3.1. Introducción

El término segmentar hace referencia, en rasgos generales, a la división de una imagen en múltiples secciones u objetos para su posterior análisis. En otras palabras, la segmentación se encarga de identificar los objetos de importancia dentro de la imagen, en este caso los marcadores. El nivel de detalle de esta división depende del problema a atacar.

Este proceso es uno de los más importantes dentro de un sistema de captura de movimiento ya que en base a los datos obtenidos aqui, se realizará tanto el tracking como la reconstrucción 3D por lo que un error en la detección de la posición de los marcadores será imposible de detectar en etapas posteriores y generará un seguimiento 3D erróneo del mismo. Más aun si se tiene en cuenta que el sistema será aplicado para la medicina, por lo que deberá tener una exactitud mayor que otros sistemas donde la precisión no juega un papel tan importante (como el Kinect de Microsoft[18] por ejemplo).

Existen varios métodos a aplicar para realizar la segmentación, en esta sección se describirán los tipos más importantes asi como la elección realizada para la implementación de este sistema, además se explicará como funciona el bloque desarrollado.

3.2. Estado del arte

Para realizar la segmentación existen varios métodos, los algoritmos para tratar imagenes monocromáticas generalmente se clasifican en dos grupos basados en la intensidad de los pixeles: discontinuidad y similaridad.

En los algoritmos basados en discontinuidad, se parte de la suposición que los límites de las regiones son suficientemente distintos unos de otros y del fondo para permitir detectarlos en base a las discontinuidades de intensidad. Los algoritmos principales en esta categoría son los basados en **detección de bordes**.

Por otro lado, en los algoritmos basados en similaridad, se busca dividir la imagen en diferentes zonas donde los pixeles de cada una son similares entre si y comparten ciertas características predefinidas. Los algoritmos más conocidos son los basados en identificación de regiones, como por ejemplo la **aplicación de umbral**.

A continuación se explican los algoritmos mencionados que, como se dijo, son dos de los más importantes dentro de la segmentación.

3.2.1. Detección de bordes

De los dos métodos nombrados anteriormente, este es el más complejo y pesado operacionalmente. Básicamente se trata de la detección de líneas o transiciones en una imágen mediante el procesamiento de los pixeles que la componen. A continuación se explica el detalle de este método, como se menciona en el libro *Digital Image Processing de Rafael Gonzalez y Richard Woods*[7].

Asi como el difuminado en una imagen (que equivale a hacer un promedio de los pixeles en una zona) puede realizarse mediante la integración, los cambios de intensidad abruptos entre pixeles continuos pueden detectarse utilizando derivadas. Por razones que serán evidentes más adelante, las derivadas de primer y segundo orden son particularmente las más indicadas para este propósito.

Las derivadas de una función digital son siempre definidas en términos de diferencia. Hay varias formas de aproximar estas diferencias pero para lograr detectar bordes de forma correcta es necesario que la aproximación usada para la derivada de primer orden:

1. valga cero en áreas de intensidad constante

2. no valga cero al inicio de un escalón o rampa de intensidad

3. no valga cero en los puntos pertenecientes a una rampa de intensidad

De la misma forma, se requiere que la aproximación utilizada para la derivada de segundo orden:

1. valga cero en áreas de intensidad constante

2. no valga cero al inicio y al final de un escalón o rampa de intensidad

3. no valga cero en los puntos pertenecientes a una rampa de intensidad

Utilizando la definición de derivada desde el punto de vista del cálculo funcional, y simplificando por el desarrollo de Taylor de primer grado alrededor del punto x, considerando una variación de ∂x unitaria, se obtiene:

$$\frac{\partial f}{\partial x} = f'(x) = f(x+1) - f(x) \tag{1}$$

análogamente, para la derivada segunda:

$$\frac{\partial^2 f}{\partial x^2} = f''(x) = f(x+1) - f(x-1) - 2f(x) \tag{2}$$

Se puede ver fácilmente que tanto la ecuación 1 como la ecuación 2 satisfacen las condiciones planteadas anteriormente. Además, considerando las propiedades de estas derivadas se puede concluir que la de primer orden es la más adecuada para detectar bordes más "gruesos" y la segunda para detectar los más finos. Asi mismo para detectar puntos aislados la más adecuada es la derivada segunda, lo que no es de sorprender ya que la misma es más sensible que la primera frente a cambios bruscos de intensidad. A raiz de esto, también se concluye que la derivada segunda es la más adecuada para detectar detalles finos (incluído el ruido). Tambien, es de destacar que mediante el signo de la la derivada segunda se puede detectar si la transición en un borde (ya sea rampa o escalón) es de luz a oscuridad o viceversa.

Por otro lado, para realizar el procesamiento de las imágenes, se analizan las mismas como matrices numéricas. Una imagen en color, se traduce como tres matrices bidimensionales, una por cada componente cromática (por ejemplo rojo, verde, azul) siendo las filas y columnas de las matrices, el ancho y largo de la imagen. A efectos de simplificar el análisis, se estudian imágenes en escala de grises lo cual implica trabajar con una sola matriz en vez de tres. En el formato de archivo de imagen TIFF, la escala de grises en 8 bits va de 0 (negro), a 255 (blanco), para cada pixel de la imagen (ver figura 1).

0	0	0	70	70	70
0	0	0	70	70	70
0	0	0	70	70	70
150	150	150	255	255	255
150	150	150	255	255	255
150	150	150	255	255	255

Figura 1: Imagen en escala de grises y su representación matricial.

La herramienta elegida para encontrar tanto la magnitud como la dirección de un borde en la posición (x, y) de la imagen f es el gradiente denotado como ∇f y definido como:

$$\nabla f = grad(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{3}$$

Este vector tiene la característica geométrica de apuntar en la dirección del mayor cambio en el rango de f en la posición (x, y). La magnitud (o largo) del vector ∇f, denotada como $M(x, y)$ donde

$$M(x,y) = mag(\nabla f) = \sqrt{g_x^2 + g_y^2} \tag{4}$$

es el valor de la tasa de cambion en la dirección del vector gradiente. La dirección del vector gradiente está dada por el ángulo

$$\alpha(x,y) = tan^{-1}\left[\frac{g_y}{g_x}\right] \tag{5}$$

que es medido respecto al eje x.

Cabe destacar que tanto g_x como g_y y $M(x,y)$ son imagenes del mismo tamaño que la original creadas cuando x e y varían de forma tal de recorrer todos los pixeles de f. Asi mismo, $\alpha(x,y)$ tambien es una imagen del mismo tamaño que la original creada por la división de la imagen g_y entre la imagen g_x.

La dirección de un borde en un punto (x,y) es ortogonal a la dirección $\alpha(x,y)$ del vector gradiente en ese punto.

Por lo tanto, para detectar un borde en una imagen resta calcular el gradiente de la imagen y luego su magnitud para cada pixel. Si la superficie es uniforme, esta magnitud será nula (o muy pequeña) y si la superficie varía (por ejemplo, cuando hay un borde de por medio) el valor de la magnitud será alta.

Como se vió anteriormente, para obtener el gradiente de una imagen se requiere realizar las derivadas paciales en cada pixel de la imagen. Como se está trabajando con valores digitales, es necesario realizar una aproximación de dichas derivadasen cada punto. De la ecuación 1 se tiene que:

$$g_x = \frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y) \tag{6}$$

y

$$g_y = \frac{\partial f(x,y)}{\partial y} = f(x,y+1) - f(x,y) \tag{7}$$

Por otro lado, se puede ver que estas dos ecuaciones pueden ser implementaddas para todos los valores de x e y pertinentes mediante el filtrado de la imagen $f(x,y)$ con las máscaras de la figura 2.

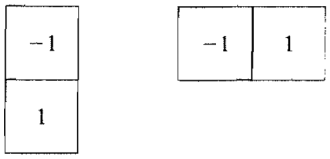


Figura 2: Máscaras de 1 dimensión para implementar ecuaciones 6 y 7.

Realizando esto se detectarán los bordes verticales y horizontales de la imagen, pero cuando es de interés detectar un borde en diagonal máscaras en 1 dimensión no funcionan, por lo que se necesita una de 2 dimensiones como las de la figura 3. Si bien las máscaras de 2x2 realizan la detección de bordes diagonales, no son tan eficientes determinando la dirección del mismo como las máscaras que son simétricas respecto al punto central, por ejemplo las de 3x3.

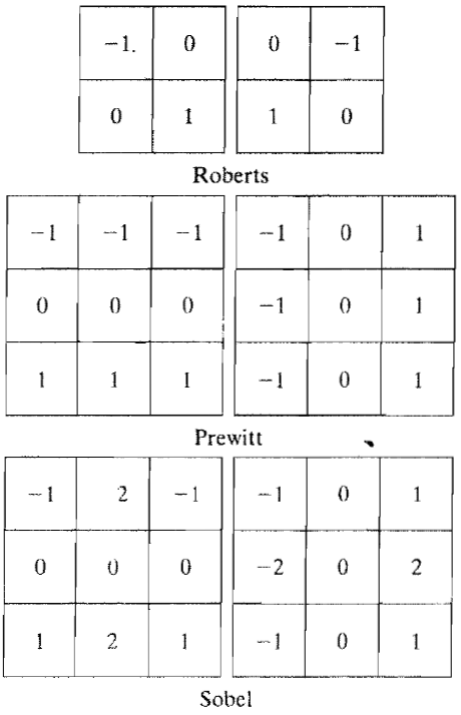


Figura 3: Máscaras de 2 dimensiones.

En la figura 4, se puede ver un ejemplo práctico de una máscara de 2 dimensiones. Para cambiar entre derivada horizontal y vertical, basta con trasponer la máscara que se utilizará en la convolución.

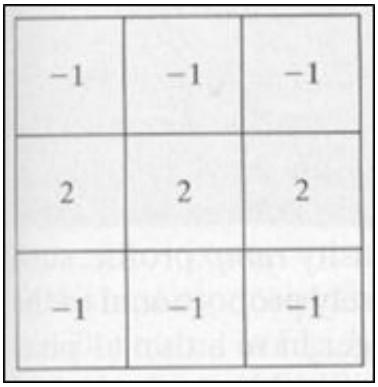


Figura 4: Ejemplo de máscara de detección de líneas.

Como se dijo anteriormente, dichas máscaras deben estar compuestas de tal forma que ante una región constante devuelva valores nulos, ya que no hay variaciones. Una forma de realizar esto es imponiendo que la suma de sus coeficientes sea nula.

Existen variedades de máscaras aplicables para esta operación, entre ellas la de Sobel (ver figura 5), que presentan beneficios adicionales como la supresión de ruido, manteniendo la característica de detectar los bordes.

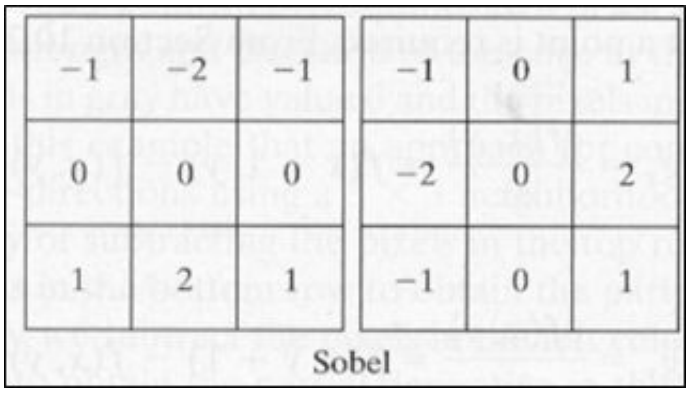


Figura 5: Máscara de Sobel.

En la figura 1 se observa una imagen de 6x6 pixeles y su representación matricial. Al aplicar la máscara de Sobel a la matriz de esta imagen, se detecta la transición entre los 4 niveles de gris mientras que se igualan los niveles constantes. En la figura 6 se pueden observar estos resultados.

0	0	255	255	0	0
0	0	255	255	0	0
255	255	255	255	255	255
255	255	255	255	255	255
0	0	255	255	0	0
0	0	255	255	0	0

Figura 6: Resultado de aplicar máscaras de Sobel sobre imagen original 6x6.

En la figura 7, se muestra un ejemplo más complejo de la detección de bordes realizada con matriz de Sobel donde se observa mejor los resultados de este método. Como se dijo antes, este método se puede combinar con otros para mejorar los resultados. Una posibilidad es someter la imagen a un proceso de *Smooth*[8] -o suavizado- previo a la detección, de esta forma se descartan los bordes pequeños (por ejemplo, los ladrillos de la casa en la figura 7a) que en general son considerados como ruido. Otra posible combinación para realizar una detección más selectiva es la aplicación de un umbral luego del cálculo del gradiente como se puede observar por ejemplo en la figura 7c. Cuando el interés recae tanto en destacar los bordes principales de una imagen como en obtener la mayor conectividad posible es común que se aplique smoothing y umbral a la vez.

(a) Imagen original.

(b) Sobel sin umbral.

(c) Sobel con umbral

Figura 7: Imagen tomada del CIPS [14]

3.2.2. Métodos de umbral

Los métodos del valor umbral son un grupo de algoritmos cuya finalidad es segmentar los objetos de una imagen en función de un rango de valores. La pertenencia de un píxel a cierto segmento se decide mediante la comparación de alguna propiedad unidimensional del mismo (por ejemplo su nivel de gris o nivel de luminosidad) con cierto valor umbral. Dado que esta comparación de valores se realiza individualmente para cada píxel, al método del valor umbral se le considera un método de segmentación orientado a píxeles.

Por lo tanto, mientras en los métodos de detección de bordes las regiones eran identificadas encontrando primero segmentos de borde y luego tratando de unir los mismos para formar bandas, en los métodos de umbral se trata de particionar la imagen directamente en regiones basándose en la intensidad de estos pixeles y/o en otras propiedades, reduciendo el problema a encontrar el umbral correcto.

En la figura 8 se observa el resultado de aplicar el método de umbral a la figura 1 con un umbral de 200.

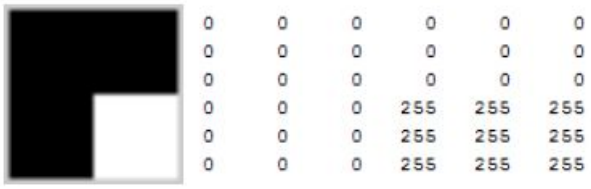


Figura 8: Resultado de aplicar un umbral de valor 200.

A continuación, se presentan algunos conceptos básicos para entender mejor la segmentación por umbral:

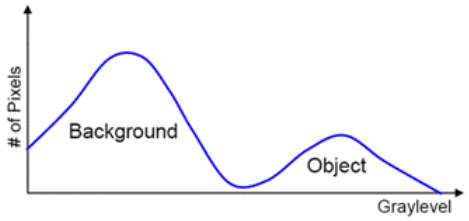


Figura 9: Histograma de intensidad de una imagen[6].

Considerando la figura 9 como el histograma de intensidad de una imagen, $f(x, y)$, se puede apreciar que

la misma está compuesta por un objeto u objetos iluminados de aproximadamente la misma intensidad y un fondo oscuro. De esta manera, se definen en este histograma sdos campanas bien determinadas. La manera más obvia de extraer los objetos del fondo es seleccionando un umbral T que separe estas dos campanas y por lo tanto cualquier punto (x, y) de la imagen que cumpla $f(x, y) > T$ será un punto perteneciente al objeto mientras que el resto son puntos pertenecientes al fondo. De acuerdo a lo anterior, la imagen segmentada puede definirse de la siguiente manera.

$$g(x,y) = \begin{cases} 1 & si \quad f(x,y) > T \\ 0 & si \quad f(x,y) \leq T \end{cases} \tag{8}$$

Si T toma un valor constante en toda la imagen entonces al proceso se le llama *umbralización global*, por otro lado si T cambia en una imagen el proceso es llamado *umbralización variable*. A veces se utiliza el término *umbralización local o regional* en la umbralización variable cuando el valor de T en un punto (x, y) depende de las propiedades de los puntos al rededor de (x, y).

En la imagen 9 se observaba un ejemplo donde se aplica el proceso más simple de umbralización sin embargo, en la mayoría de los casos ajustar el histograma de una imagen a esta forma no da tan buenos resultados. Para estas situaciones se recurre a la *umbralización múltiple*, donde un punto (x, y) se puede clasificar en varias clases dependiendo de la complejidad de la imagen.

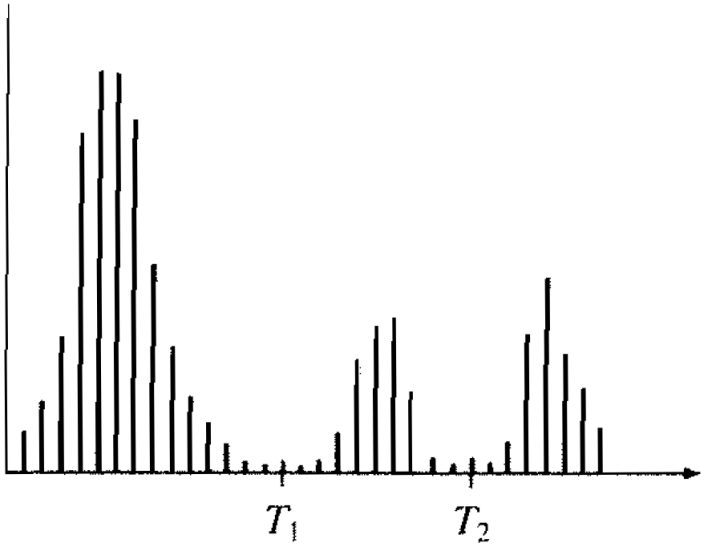


Figura 10: Histograma de intensidad de tres clases[10].

En la figura 10 se puede ver el histograma de una imagen con 3 clases dominantes correspondientes, por ejemplo, a un objeto brillante, otro un poco menos brillante y un fondo oscuro. En este caso la umbralización de 3 clases clasificará el punto (x, y) como perteneciente al fondo si $f(x, y) \leq T_1$, perteneciente a un objeto si $T_1 < f(x, y) \leq T_2$ y perteneciente al objeto más brillante si $f(x, y) > T_2$. Por lo tanto, la imagen segmentada será de la forma:

$$g(x,y) = \begin{cases} a & si \quad f(x,y) > T_2 \\ b & si \quad T_1 < f(x,y) \leq T_2 \\ c & si \quad f(x,y) \leq T_1 \end{cases} \tag{9}$$

donde a, b y c son tres valores distintos de intensidad.

Observando los histogramas anteriores, puede verse que la efectividad de la umbralización está directamente relacionada con el ancho y la profundidad de los valles que separan las distintas clases. Siguiendo esto, los factores claves que afectan directamente al tamaño de estos valles son:

- Separación entre picos: cuanto más separados, mejor posibilidad de segmentar correctamente.

- Ruido de la imagen.
- La relación entre los tamaños de los objetos y el fondo.
- La uniformidad de la iluminación.
- La uniformidad de las propiedades de reflexión de la imagen.

Es de destacar que, si bien no resulta tan evidente como pasa con el ruido de la imagen, la iluminación y las propiedades de reflexión juegan un papel clave para obtener una segmentación efectiva y por lo tanto controlar estos parámetros debe ser prioridad si se quiere obtener una buena segmentación. Cuando no es posible controlarlos, existen tres aproximaciones básicas que se pueden realizar para mejorar los resultados: corregir el patrón de sombras directamente, corregirlo mediante algun proceso ya establecido (por ejemplo utilizando la transformada top-hat[9]) o aplicar un umbral variable como se mencionó anteriormente.

Como se mencionaba anteriormente, cuando la distribución de las intensidades entre objeto y fondo se encuentran lo suficientemente distinguidas, es posible utilizar un solo umbral global aplicable en toda la imagen. Por otro lado, para aplicaciones donde el valor del umbral debe ir cambiando para una secuencia de imagenes, es recomendable utilizar algun método para calcular el valor del mismo automáticamente. En base a los factores planteados anteriormente que afectan la imagen y a las restantes características de la misma, se han implementado distintas formas de obtener el valor de umbral. El siguiente algoritmo iterativo muestra un ejemplo sencillo de como calcular el umbral:

1. Seleccionar un umbral inicial T estimado.
2. Segmentar la imagen utilizando T. Esto producirá dos conjuntos de pixeles, los que estén por encima del umbral (C_1), y los que estén por debajo (C_2).
3. Calcular el promedio de las intensidades (m_1 y m_2) de los pixeles en C_1 y C_2 respectivamente.
4. Calcular un nuevo umbral $T = \frac{1}{2}(m_1 + m_2)$.
5. Repetir los pasos 2 a 4 hasta que la diferencia entre los umbrales T de sucesivas iteraciones sea menor a un ΔT definido anteriormente.

Umbral de Otsu

Los principales métodos existentes para obtener el valor del umbral están listados en el survey de Mehmet Sezgin[5], en el cual se clasifica estos métodos en las siguientes categorías:

- Basados en la forma del histograma.
- Basados en agrupamiento.
- Basados en la entropía de las regiones.
- Basados en los atributos de los objetos.
- Espaciales.
- Locales.

Entre los cuarenta métodos exhibidos en este paper, se encuentra el método de Otsu[11]. El mismo se encuentra dentro de los métodos basados en agrupamiento y es uno de los más utilizados en segmentación por umbral debido a su eficacia y simplicidad. Utiliza técnicas estadísticas para resolver el problema. En particular se utiliza la varianza que, como es sabido, es una medida de la dispersión de valores (en este caso se trata de la dispersión de los niveles de gris).

El método de Otsu[11] calcula el valor umbral de forma que la dispersión dentro de cada segmento sea lo más pequeña posible, pero al mismo tiempo sea lo más alta posible entre segmentos diferentes. Para ello

se calcula el cociente entre ambas varianzas (para el caso de dos clases) y se busca un valor umbral para que este cociente sea máximo.

Dicho de otra manera, se puede ver al proceso de umbralización como un problema estadísitico cuyo objetivo es minimizar el error promedio que se produce al asignar los pixeles de la imagen a dos o más clases. La solución a este problema es conocida como *regla de decisión de Bayes[13]*, sin embargo aplicar esta regla no es tan sencillo ya que estimar la densidad de probabilidad de cada clase no es simple. El método de Otsu[11] es considerado una de las mejores aproximaciones a esta solución, ya que maximiza la "varianza intermedia entre clases" (*between class variance*[1]) que es una medida muy utilizada en problemas de discriminación estática lo que permite obtener un umbral óptimo. A esto se le suma la ventaja de que todos los cálculos realizados en el método se realizan sobre el histograma de intensidades que es muy fácil de obtener.

La "varianza intermedia entre clases" puede escribirse como

$$\sigma_B^2 = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 = P_1P_2(m_1 - m_2)^2 = \frac{(m_GP_1 - m)^2}{P_1(1 - P_1)} \tag{10}$$

donde P_1 y P_2 son las probabilidades de que un pixel sea asignado a la clase 1 y 2 respectivamente, m_1 y m_2 son las medias de las intensidades de cada una de estas clases. Además, $m(k)$ es la media (intensidad promedio) acumulada hasta el nivel k y m_G es la intensidad media (intensidad global promedio) de la imagen en su totalidad:

$$m(k) = \sum_{i=0}^{k} ip_i \tag{11}$$

$$m_G(k) = \sum_{i=0}^{L-1} ip_i \tag{12}$$

y considerando que k es el umbral que separa la clase 1 de la clase 2, P_1 puede escribirse como

$$P_1(k) = \sum_{i=0}^{k} p_i \tag{13}$$

donde p_i es la cantidad normalizada de pixeles de la imagen que tienen intensidad i.
Por lo que la ecuación 10 tambien queda dependiendo del umbral k:

$$\sigma_B^2(k) = \frac{(m_GP_1(k) - m(k))^2}{P_1(k)(1 - P_1(k))} \tag{14}$$

Para el caso de umbralización con múltiples clases (K clases), la varianza intermedia vale:

$$\sigma_B^2(k) = \sum_{k=1}^{K} P_k(m_k - m_G)^2 \tag{15}$$

donde

$$P_k = \sum_{i \epsilon C_k} P_i$$

$$m_k = \frac{1}{P_k} \sum_{i \epsilon C_k} ip_i$$

y m_G es la ganancia global como se definió anteriormente. Esta umbralización implica tener $K-1$ umbrales.

A modo de ejemplo, para 3 clases (3 niveles de intensidades separadas por 2 umbrales) la "varianza intermedia entre clases" queda:

$$\sigma_B^2(k) = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 + P_3(m_3 - m_G)^2 \tag{16}$$

[1] diferencia entre la varianza total y la suma de las varianzas de cada clase[17]

donde

$$P_1 = \sum_{i=0}^{k_1} p_i$$

$$P_2 = \sum_{i=k_1+1}^{k_2} p_i$$

$$P_3 = \sum_{i=k_2+1}^{L-1} p_i$$

y

$$m_1 = \frac{1}{P_1} \sum_{i=0}^{k_1} i p_i$$

$$m_2 = \frac{1}{P_2} \sum_{i=k_1+1}^{k_2} i p_i$$

$$m_3 = \frac{1}{P_3} \sum_{i=k_2+1}^{L-1} i p_i$$

Además, como en el caso de 2 clases, se dan la siguientes relaciones:

$$P_1 m_1 + P_2 m_2 + P_3 m_3 = m_G \tag{17}$$

y

$$P_1 + P_2 + P_3 = 1 \tag{18}$$

Luego, aplicando lo visto anteriormente acerca del umbral de Otsu, se tiene que el umbral óptimo k^* es el valor de k que maximiza 14 (para el caso de múltiples clases, serían los valores de k_k^* que maximizan 15). Para encontar k^* basta con evaluar la ecuación 14 para todos los valores de k válidos[2] y seleccionar el valor de k que maximiza dicha ecuación. Si el máximo $\sigma_B^2(k)$ se da para varios k, k^* se calcula como el promedio de los k que dan dicho valor.

Para el ejemplo del algoritmo de 3 clases, se deberían encontrar los valores de k_1 y k_2 que maximicen la varianza entre clases. Para ello, se evalúa la ecuación 15 para todos los pares (k_1, k_2) posibles, es decir: $(k_1, k_2) tq 0 < k_1 < k_2 < L - 1$.

Algo importante a destacar es que este método es poco costoso en términos computacionales ya que el máximo numero de $k's$ para los que hay que evaluar la ecuación 14 es L, que corresponde a la cantidad de niveles de intensidad de la imagen.

En resumen, el *algoritmo de Otsu* se puede implementar de la siguiente manera:

1. Realizar el histograma de la imagen, donde cada componente corresponde a un nivel de intensidad (con un total de L niveles)
2. Calcular la probabilidad $P_1(k)$ con la ecuación 13 para $k = 0, 1, 2, ..., L - 1$.
3. Calcular la media $m(k)$ con la ecuacion 11 para $k = 0, 1, 2, ..., L - 1$.
4. Calcular la media global m_G con la ecuación 12.
5. Calcular la "varianza intermedia entre clases" (*between-class variance*), $\sigma_B^2(k)$, como se muestra en la ecuación 14 (o 15) para $k = 0, 1, 2, ..., L - 1$.
6. A partir del punto anterior, obtener el umbral de Otsu k^* como el valor de k (o los valores de k_k para el caso de múltiples clases) que maximiza $\sigma_B^2(k)$.

[2] todos los k enteros tal que $0 \leq k \leq L - 1$ (con $L - 1$ nivel de intensidad máximo de la imagen) que verifiquen $0 < P_1(k) < 1$

3.3. Justificación y explicación del algoritmo

Debido a sus propiedades intuitivas, simplicidad en la implementación y a su rapidez computacional, para este sistema se eligió utilizar un método de umbral. En particular se eligió el método de Otsu[11] de tres clases ya que ofrece un buen compromiso entre simplicidad y eficacia.

Para realizar esta elección se tuvo en cuenta que las capturas a procesar serán realizadas en un ambiente controlado por lo que no es necesario implementar un método de mayor complejidad que sea más robusto frente a ciertos tipos de ruidos o características que se pueden dar en otro tipo de capturas (iluminación, fondo, traje del paciente, etc.). Como se explicó anteriormente, a partir del histograma de la imagen se pretende separar los pixeles de la imagen en 3 niveles y encontrar dos umbrales que los separen. Dado que en las capturas a procesar se tendrán marcadores blancos y el resto de la imagen lo más oscura posible, el umbral definitivo para la segmentación será el más alto de los dos obtenidos. Trabajar con tres clases permite obtener mejores resultados que al trabajar con dos ya que separa los pixeles en un nivel más y por lo tanto el umbral de intensidades calculado tendrá mayor exactitud. Esto permite ser un poco más flexible con los contrastes entre los marcadores y el resto de la imagen por lo que no sería estrictamente necesario, por ejemplo, que el traje del paciente y el fondo sean del mismo color (ver figura 13a).

El bloque de segmentación de este sistema fue implementado en el lenguaje C++, utilizando las librerías de procesamiento de imágenes OpenCV[15] y CVBlob[16].

3.3.1. Algoritmo

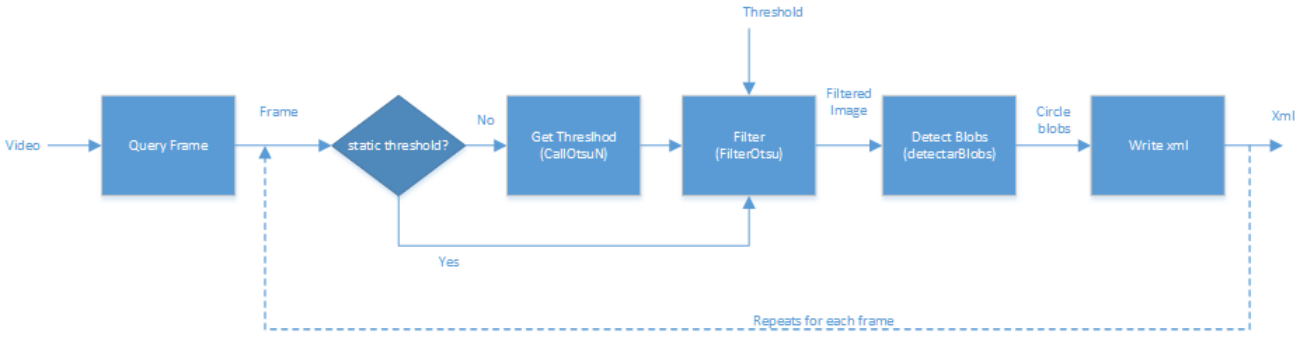


Figura 11: Diagrama de flujo del algoritmo de segmentación.

En la figura 11 se presenta un diagrama donde se observa el flujo del algoritmo de segmentación realizado. Los nombres que aparecen entre paréntesis dentro de algunos bloques son los nombres de las funciones dentro del código que implementan cada bloque.

El algoritmo realiza la segmentación a través de los siguientes pasos:

1. Se recibe como entrada un video y este es separado en cada uno de sus cuadros a través del bloque *Query Frame*.
2. Se toma un cuadro, y se calcula el umbral de Otsu con el bloque *Get Threshold*. Si al comenzar la segmentación es ingresado un umbral fijo, este paso se saltea.
3. Con el umbral calculado (o ingresado), se filtra el cuadro en el bloque *Filter*.
4. A partir de la imagen filtrada, se identifican los marcadores con el bloque *Detect blobs*.
5. Se escribe la posición de los marcadores detectados para este cuadro en un archivo con formato xml.
6. Se toma el siguiente cuadro, y se repite el proceso a partir del paso 2.

El bloque *Query Frame* es implementado mediante las funciones *cvCaptureFromAVI* y *cvQueryFrame*, las cuales pertenecen a la librería *OpenCV*[15].

Por otro lado, el bloque *Get Threshold* contiene una implementación del algoritmo de Otsu[11] de N clases[12], cedida por Matias Tailanian y Juan Cardelino, que es utilizada con $N = 3$. Como se vió en la sección 3.2.2, este método devuelve 2 umbrales de los cuales se tomará el mayor de ellos, dado que las hipótesis del problema establecen que la adquisición de video debe realizarse sobre fondo oscuro y con el paciente utilizando ropa oscura, de forma tal que los marcadores sean los elementos más claros en la imagen.

En la figura 12, se observa un diagrama que describe el funcionamiento del bloque *Filter*. Este bloque es el encargado de filtrar la imagen segun la intensidad de los pixeles y está implementado por la función *FilterOtsu*, que recibe como parámetros de entrada una imagen (uno de los cuadros de la secuencia) y el umbral a utilizar para el filtrado. Primero se le aplica a la imagen un difuminado (*smoothing*) con un filtro de mediana, con el objetivo de reducir el ruido. Luego se cambia el el espacio de colores de la imagen de RGB a HSV ya que este último es el más adecuado para realizar segmentación basada en la intensidad de los pixeles[20]. Por último, se filtra la imagen con le umbral ingresado utlizando la función *cvThreshold* de la librería OpenCV[15]. Esta función compara la intensidad de cada pixel de la imagen con el valor del umbral estableciendo un nuevo valor para la intensidad: 0[3] para los pixeles que originalmente tenían intensidad menor al umbral y 255[4] para los que originalmente presentaban intensidad mayor.

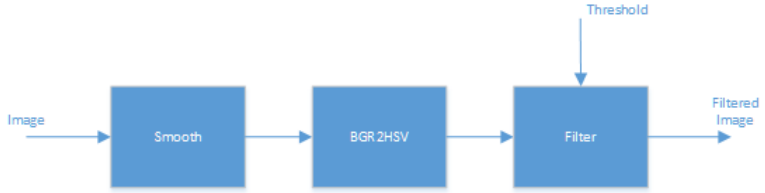
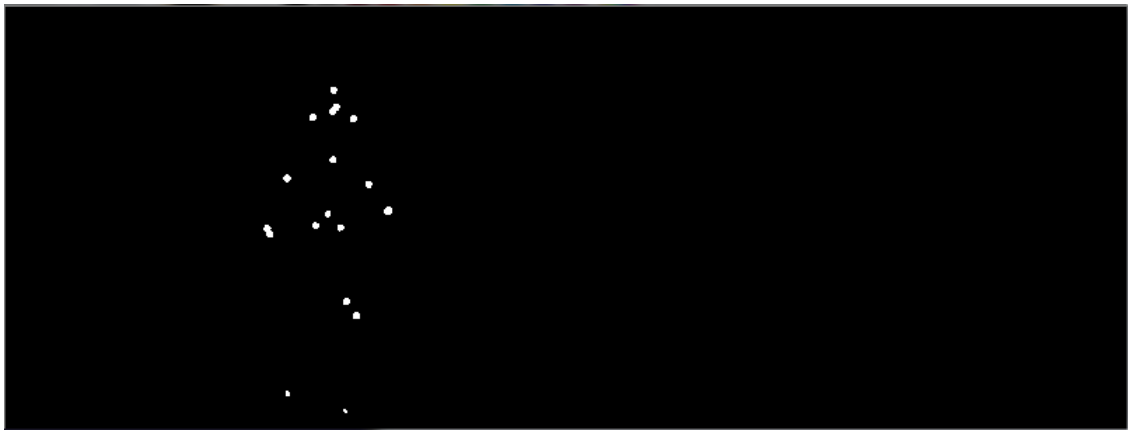


Figura 12: Diagrama de flujo del bloque de umbralización.

En la figura 13, se puede ver un ejemplo de los resultados de aplicar este bloque a una imagen sintética de la base de datos. Se puede ver que la intensidad de los pixeles azules y negros queda por debajo del umbral, mientras que los pixeles blancos quedan por encima.

[3] negro en el espacio HSV.
[4] blanco en el espacio HSV.

(a) Captura original de una secuencia sintética.



(b) Imagen filtrada con el umbral de Otsu.

Figura 13: Entrada y salida del bloque de umbralización.

El funcionamiento del bloque *Detect blobs* es descrito por el diagrama de la figura 14. Este bloque recibe como entrada la imagen previamente filtrada por el bloque *Filter* y da como salida una imagen con los marcadores detectados e identificados. En primer lugar, se identifican todos los blobs[5] de la imagen filtrada con el bloque *Find blobs*, que es implementado por la función *cvLabel* de la librería CVBlobs[16]. Cuando se dice "identificar todos los blobs", básicamente se hace referencia a identificar cada grupo de pixeles blancos continuos de la imagen filtrada como un objeto (un blob) único. Luego, si se ingresó la opción para filtrar por área, los blobs[**?**] detectados se filtran por area máxima y/o mínima mediante la función *cvFilterByArea* perteneciente a la librería CVBlobs[16].

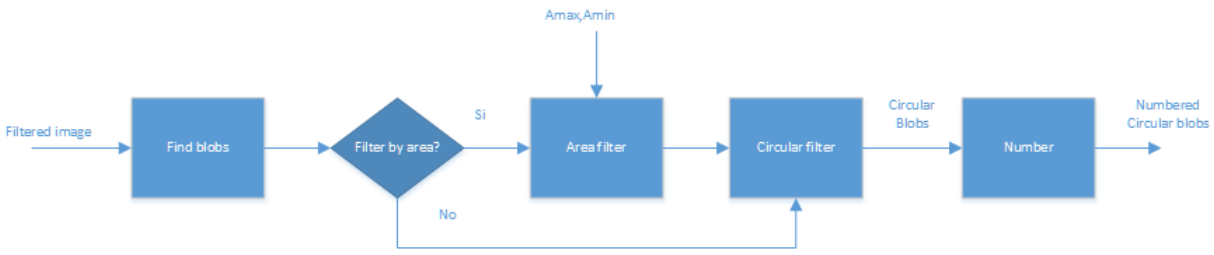


Figura 14: Diagrama de flujo del bloque de detección de blobs.

Ya sea que se haya filtrado por área o no, la imagen con blobs pasa por el bloque *Circular filter* donde se

[5] Binary Large Objects[21]

descartan los blobs que no tienen forma circular. Para ello, se usan dos propiedades de los blobs: momentos y excentricidad.

Finalmente, el bloque *Write xml* es implementado mediante las funciones de $C++$ para escribir archivos, teniendo en cuenta la estructura de los archivos xml[19].

Luego de importar el video, realiza el cálculo de umbral de Otsu[11] de tres niveles para cada cuadro, y luego obtiene los pixeles que se encuentran por encima de este umbral (ver figura 13b).

En el caso ideal, estos pixeles corresponden a los marcadores en el paciente. En la práctica, no todos los pixeles detectados corresponden a marcadores, por lo que luego de obtenidos los mismos, se detectan los objetos (conjuntos de pixeles detectados que están contiguos, ver figura 15) y se filtran los mismos según su área y su excentricidad obteniendo finalmente sólo los objetos de forma circular y de determinada área (ver figura 16).

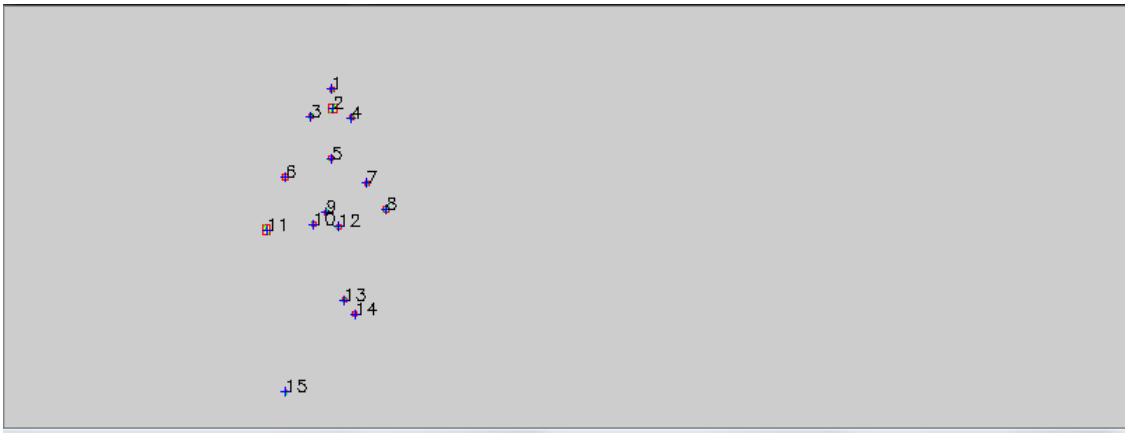


Figura 15: Objetos detectados.

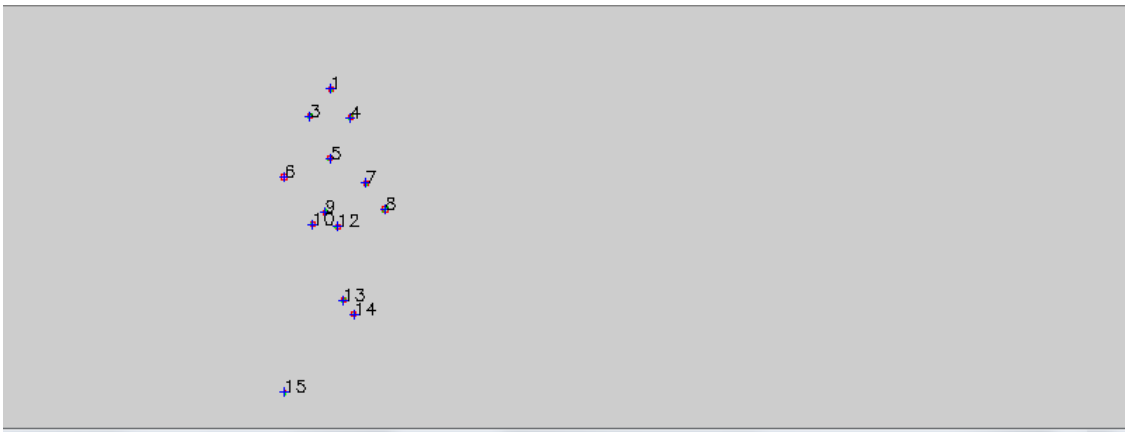


Figura 16: Filtro de objetos circulares y de determinada área.

Finalmente, es de destacar que la salida de este bloque se expone en un archivo en formato xml, que contiene la posición del centroide de cada marcador, para cada cuadro. Se eligió el formato xml para exportar los resultados ya que es un formato conocido universalmente y fácil de importar en cualquier lenguaje, en particular Matlab contiene librerías para trabajar con el mismo.

3.4. Resultados y análisis

Queda pendiente como posible mejora a futuro robustecer este bloque, de forma tal de poder detectar marcadores en otros contextos no tan amigables como las condiciones del laboratorio.

4. Discusión y análisis de resultados

5. Conclusiones

Referencias

[1] Herda L., Fua P., Plankers R., Boulic R. Using skeleton-based tracking to increase the reliability of optical motion capture.Human movement science, 20(3), 313-341, 2001.

[2] Malik N., Dracos T. & Papantoniou D. (1993). Particle tracking in three-dimensional turbulent flows - Part II: Particle tracking, Experiments in Fluids, 15:279–294.

[3] Faugeras o., Robert L. What can two images tell us about a third one. International Journal of Computer vision, 18(1), 485-492,1996.

[4] C.A.Diaz, M.L.Toro, J.C.Forero, A.Torres Detección, rastreo y reconstrucción tridimensional de marcadores pasivos para análisis de movimiento humano. Cinemed iii Revista ingeniería Biomédica issn 1909-9762, volumen 3, número 6, julio-diciembre 2009, págs. 56-67 Escuela de Ingeniería de Antioquia-Universidad ces, Medellín, Colombia

[5] M.Sezgin,B.Sankur, "Survey over image thresholding techniques and quantitative performance evaluation", Journal of Electronic Imaging 13(1), 146–165 (January 2004).

[6] http://deploy.virtuallabs.ac.in/labs/cse19/theory.php?exp=segment

[7] Digital Image Processing. Rafael Gonzalez & Richard Woods.

[8] pág. 174, sección 3.5 Digital Image Processing. Rafael Gonzalez & Richard Woods.

[9] pág. 692, sección 9.6.3 Digital Image Processing. Rafael Gonzalez & Richard Woods.

[10] pág. 760, sección 10.3 Digital Image Processing. Rafael Gonzalez & Richard Woods.

[11] Nobuyuki Otsu (1979). .A threshold selection method from gray-level histograms". IEEE Trans. Sys., Man., Cyber.

[12] Matías Tailanian, Juan Cardelino. https://github.com/martin-etchart/kde 24/11/2014.

[13] pág. 894, sección 12.2.2 Digital Image Processing. Rafael Gonzalez & Richard Woods.

[14] D. Phillips; "Image Processing in C: Analyzing and Enhancing Digital Images", RandD Publications, 1994

[15] http://opencv.org/

[16] https://code.google.com/p/cvblob/

[17] sección 4.4.4, http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MORSE/threshold.pdf

[18] http://www.xbox.com/es-ES/Kinect

[19] http://es.wikipedia.org/wiki/Extensible_Markup_Language

[20] Shamik Sural, Gang Qian and Sakti Pramanik (2002). "Segmentation And Histogram Generation Using The HSV Color Space For Image Retrieval". Michigan State University, USA.

[21] http://en.wikipedia.org/wiki/Binary_large_object