UMBRALIZACIÓN MÚLTIPLE UTILIZANDO EL MÉTODO DE OTSU PARA RECONOCER LA LUZ ROJA EN SEMÁFOROS

Pedro Huamaní Navarrete

RESUMEN

Este artículo presenta un enfoque práctico de la implementación algorítmica del método de Otsu en el software Matlab, con la finalidad de determinar un umbral múltiple en imágenes de color. De esta forma, se reconoce en imágenes capturadas de varios puntos de la ciudad durante el día, la luz roja mostrada por los semáforos. Para ello, se determinó tres umbrales en las imágenes de cada canal de color utilizando el método de Otsu, seguido de una operación lógica de intersección, para luego complementarlo con una operación morfológica de erosión y dilatación. Fue así que se logró identificar el semáforo con la luz roja encendida, permitiendo de esta manera proporcionar una ayuda al conductor en un momento de distracción. La técnica empleada fue aplicada a un total de 20 imágenes de diferentes lugares de la ciudad de Lima, y originó un error del 5% en el reconocimiento de la luz roja.

Palabras clave: Método de Otsu, umbralización múltiple, operadores morfológicos.

USING MULTIPLE THRESHOLDING OTSU METHOD TO RECOGNIZE THE RED LIGHT ON TRAFFIC LIGHTS

ABSTRACT

This paper presents a practical approach to implementing the algorithm Otsu method in Matlab software, in order to determine a threshold in multiple color images. Thus, it is recognized in captured images of several points in the city during the day, the red light images shown by the traffic lights. For this, three thresholds were determined on images of each color channel using Otsu's method, followed by a logical operation intersection, and then complemented with a morphological erosion and dilation operation. It was so the light was identified with the red light, thereby allowing the driver to provide aid in a moment of distraction. The technique was applied to a total of 20 images from different places of the city of Lima, and causing an error of 5% in recognition of the red light.

Keywords: Otsu's method, thresholding multiple, dilate morphological operators.

Recibido: 07/09/2015 Aprobado: 30/10/2015

Introducción

El progreso de la tecnología digital permite, en la actualidad, utilizar métodos y técnicas matemáticas de reconocimiento de patrones, sea en imágenes estáticas o en dinámicas. Sin embargo, existe una diversidad de estas técnicas, y donde muchas de ellas se adaptan a problemas muy particulares.

Por lo cual, es necesario poner en práctica las técnicas matemáticas, eligiéndolas y combinándolas adecuadamente para cumplir el propósito del reconocimiento de patrones. No obstante, dicha operación amerita el uso de software de simulación con alto grado de procesamiento para computación numérica. Y es por tal razón que el software Matlab fue elegido para realizar tal procedimiento.

Si bien es cierto que el reconocimiento de patrones tiene una principal aplicación en el control de calidad, este también puede ser ampliamente utilizado en la parte comercial, y particularmente en la ayuda al conductor. Como es sabido, muchos conductores y especialmente aquellos que realizan la tarea de conducir vehículos de transporte interurbano, deben de encontrarse en alerta ante cualquier situación. Es así que, una cámara fotográfica sumada a un procesador, pueden hacer posible la disminución del número de accidentes que se presentan día a día, cuando el conductor del vehículo no logra divisar la luz roja del semáforo en el momento adecuado.

De esta forma, en este artículo se pretende otorgar las bases matemáticas para la segmentación de la luz roja del semáforo, capturado a través de una fotografía en diferentes puntos de la ciudad. Por lo cual, se utilizó el método de Otsu para determinar un número mayor de umbrales, que permitió facilitar la identificación de un patrón en particular.

Metodología

La metodología empleada para el desarrollo de este trabajo implicó la participación de cuatro etapas principales. Primero, la captura de imágenes reales propias de una ciudad en estado diurno. Segundo, la aplicación de la técnica de Otsu para determinar un umbral múltiple. Luego, una operación lógica de intersección con las imágenes de cada canal de color, para finalizar con una operación morfológica de dilatación y/o erosión. A continuación se detalla cada una de las etapas desarrolladas.

1. Captura de imágenes

Esta etapa consistió en la captura de imágenes de escenarios reales en puntos distintos de la ciudad de Lima, con la presencia de un semáforo vehicular en el momento que la luz roja se encontraba activada. Es necesario resaltar, que esta captura fue realizada con un ángulo de elevación aproximado de 60° con respecto al nivel del suelo, con la finalidad

de simular la captura de imagen por medio de una cámara fotográfica situada en la parte delantera de un auto convencional. Además, todas las capturas fueron realizadas durante el día y con presencia de luz solar.

El número de capturas de imágenes fue igual a 20, todas ellas a colores con resolución de 192x256x3 pixeles y codificación de 24 bits/pixel. Por lo cual, tal como se observa en la figura 01, algunas de las imágenes presentan un fondo irregular con alta variedad de colores de vehículos y de edificios con colores similares al rojo. Mientras que otras imágenes presentan cierta cantidad de vegetación y un cielo claro que facilita discriminar el color de interés. Por lo cual, la gran variedad de colores en cada una de las imágenes capturadas, trajo como consecuencia una dificultad para realizar la segmentación a partir de un valor numérico de un pixel para un determinado color. Por tal razón, se procedió a una operación lógica binaria y condicional entre las imágenes de cada canal de color, complementado con una determinación de un número mayor de umbrales en cada una de estas imágenes con la finalidad de facilitar tal segmentación.

Figura 01. Imagen capturada para cuatro puntos diferentes de la ciudad.

2. Separación de canales de color

Como las imágenes capturadas se encuentran en un formato de color, fue necesario realizar la separación de cada canal para facilitar el procesamiento y así determinar el color de interés. Para ello, se procedió a utilizar algunas funciones del Toolbox Image Processing del software Matlab. Primero, para leer la imagen con extensión JPG, luego para realizar la separación por canal de color y por último visualizarlo en una ventana en formato de gris para identificar la o las zonas con tendencia al color rojo. A continuación, las figuras 02 y 03 muestran el resultado de la separación por canal de color para las dos primeras imágenes de la figura 01, respectivamente.

```
>> A1 = imread('semaforo1.jpg');
>> A2 = imread('semaforo2.jpg');
>> A3 = imread('semaforo3.jpg');
>> A4 = imread('semaforo4.jpg');

>> A1r = A1( : , : , 1);          A1g = A1( : , : , 2);          A1b = A1( : , : , 3);
>> A2r = A2( : , : , 1);          A2g = A2( : , : , 2);          A2b = A2( : , : , 3);

>> colormap( gray(256))
>> subplot(221), image( A1 ),     subplot(222), image( A1r )
>> subplot(223), image( A1g ),    subplot(224), image( A1b )

>> colormap(gray(256))
>> subplot(221), image( A2 ),     subplot(222), image( A2r )
>> subplot(223), image( A2g ),    subplot(224), image( A2b )
```

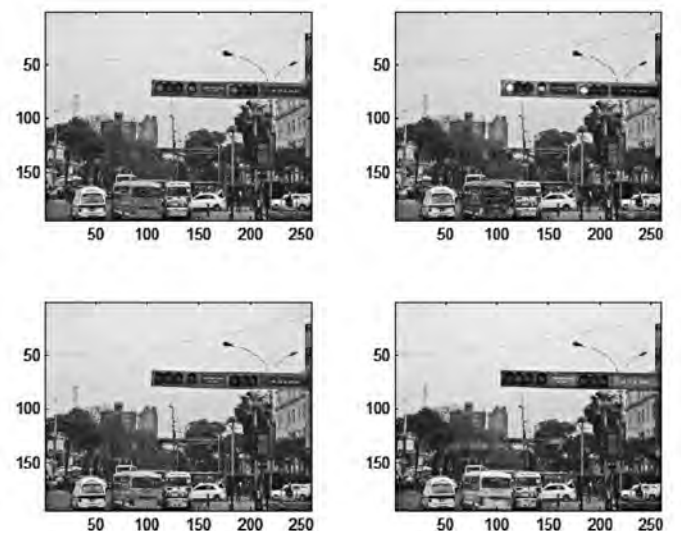

Figura 02. Primera Imagen de color y sus tres canales de color: Red, Green y Blue.

Tal como se puede apreciar en las figuras 02 y 03, existe presencia de objetos de color próximo al rojo localizados en la parte inferior de la imagen, y por lo cual también pueden ser identificados con la técnica planteada. Tal es el caso de un semáforo peatonal u otro tipo de objeto que proporciona el color rojo, y por lo tanto es necesario retirarlo. Entonces, bajo la premisa que el semáforo siempre será del tipo aéreo y por lo cual localizado en la parte superior de la imagen, se procedió a analizar solamente la mitad de filas de la imagen original. Esto quiere decir, que solamente se trabajó con las primeras 96 filas pero si con todas las columnas. Ver la figura 04.

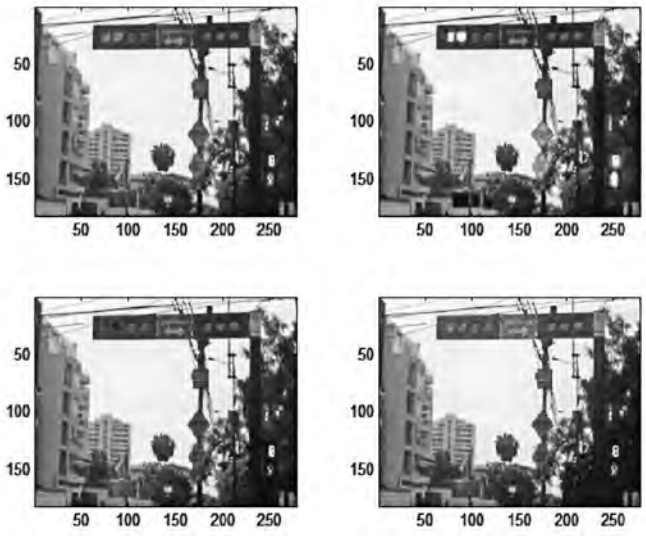

Figura 03. Segunda Imagen de color y sus tres canales de color: Red, Green y Blue.

Figura 04. Segunda Imagen de color seccionada y sus correspondientes canales de color: Red, Green y Blue.

3. Umbralización Múltiple

Asimismo, de la figura 04 se puede observar que de las tres imágenes en tonos de gris, una de ellas presenta una zona blanca de pixeles que coincide con la luz roja de la imagen a color original. Esto demuestra que las otras dos imágenes de tonos de gris corresponden a los canales de color: verde y azul, respectivamente.

Por tal razón, se optó por aplicar el histograma solamente a la imagen de gris correspondiente al canal rojo, con la finalidad de determinar un umbral que permita binarizar

dicha imagen. Sin embargo, tal como se aprecia en el histograma de la figura 05, se hace imposible determinar eficazmente el umbral adecuado.

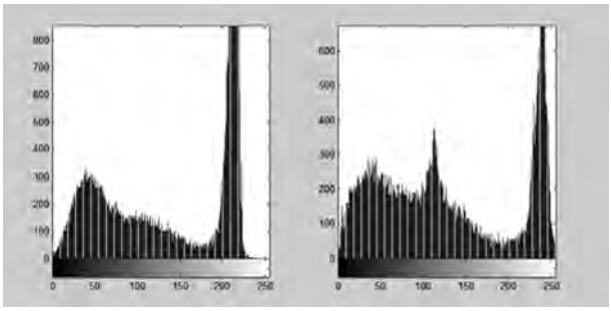

Figura 05. Resultado del histograma para cada imagen del canal de color: rojo.

Método de Otsu [Gonzalez, Woods y Eddins, 2004]

La formulación matemática de este método está basado en el histograma. Por lo cual, normaliza dicho histograma como una función de densidad de probabilidad del tipo discreta P_r. Donde, N es el número total de pixeles en la imagen, n_q es el número de pixeles que tienen un nivel de intensidad r_q, y L es el número total de posibles niveles de intensidad en la imagen.

$$P_r(r_q) = \frac{n_q}{N}$$
$$q = 0,1,2,\ldots, L-1 \qquad (01)$$

Una vez normalizado el histograma, se elige un umbral k tal que C_0 represente a un conjunto de pixeles con niveles {0, 1, 2, ..., k-1} y C_1 otro grupo de pixeles con niveles {k, k+1, k+2, …, L-1}. Por lo cual, el método de Otsu selecciona el valor del umbral maximizando la varianza σ_B^2, la cual se encuentra definida por:

$$\sigma_B^2 = w_0 * (\mu_0 - \mu_T)^2 + w_1 * (\mu_1 - \mu_T)^2 \qquad (02)$$

Y, donde:

$$w_0 = \sum_{q=0}^{k-1} P_q(r_q)$$
$$w_1 = \sum_{q=k}^{L-1} P_q(r_q) \qquad (03)$$

$$\mu_0 = \sum_{q=0}^{k-1} q * \frac{P_q(r_q)}{w_0}$$
$$\mu_1 = \sum_{q=k}^{L-1} q * \frac{P_q(r_q)}{w_1} \qquad (04)$$

$$\mu_T = \sum_{q=0}^{L-1} q * P_q(r_q)$$

Y, para el caso de umbralizar un número de M niveles, el método de Otsu se generaliza obteniendo el conjunto multinivel que maximiza la varianza entre las M clases $\{K_1, K_2, K_3, \ldots, K_M\}$, de tal forma que:

$$\{K_1, K_2, K_3, \ldots, K_M\} = MAX\{\ \sigma_B^2(k_1, k_2, k_3, \ldots, k_M)\ \}$$
$$1 \le k_1 < k_2 < k_3 < k_M < L \qquad (05)$$

Donde:

$$\sigma_B^2 = \sum_{k=1}^{M+1} w_k * (\mu_k - \mu_T)^2 \qquad (06)$$

$$w_k = \sum_{q \in C_k} P_q \qquad (07)$$
$$\mu_k = \sum_{q \in C_k} \frac{q * P_q}{w_k}$$

Para esta investigación se eligió un número de 4 clases, correspondiendo a 3 umbrales. Sin embargo, a continuación solo se muestra el procedimiento del método de Otsu para un caso de solo 1 umbral. Posteriormente, este algoritmo fue generalizado para el caso de interés, es decir 3 umbrales.

```
>> M = 1;
>> A2 = A1;
>> for i=0:255;
[x , y] = find( A1(: , : , 1) == i ); Pr(i+1) = length(x) / 192 / 256;
[x , y] = find( A1(: , : , 2) == i ); Pg(i+1) = length(x) / 192 / 256;
[x , y] = find( A1(: , : , 3) == i ); Pb(i+1) = length(x) / 192 / 256;
end

>> for k = 1:255;
for i = 1:k
      w1r = sum(Pr(1 : i)); w1g=sum(Pg(1 : i)); w1b=sum(Pb(1 : i));
      u1r = sum( [1:i] .* Pr(1:i)/w1r );
      u1g = sum( [1:i] .* Pg(1:i)/w1g );
      u1b = sum( [1:i] .* Pb(1:i)/w1b );
end
for i = k+1:255
      w2r = sum(Pr(i+1:255)); w2g=sum(Pg(i+1:255)); w2b = sum(Pb(i+1:255));
```

```
        u2r = sum( [i+1:255] .* Pr(i+1:255)/w2r );
        u2g = sum( [i+1:255] .* Pg(i+1:255)/w2g );
        u2b = sum( [i+1:255] .* Pb(i+1:255)/w2b );
    end
    uTr = w1r * u1r + w2r * u2r;
    uTg = w1g * u1g + w2r * u2g ;
    uTb= w1b * u1b + w2r * u2b;
    maxVARr(k) = w1r*(u1r -uTr)^2 + w2r*(u2r -uTr)^2;
    maxVARg(k) = w1g*(u1g -uTg)^2 + w2g*(u2g -uTg)^2;
    maxVARb(k) = w1b*(u1b -uTb)^2 + w2b*(u2b -uTb)^2;
     end

    >> [ a , umbralR ] = max( maxVARr );
    >> [ a , umbralG ] = max( maxVARg );
    >> [ a , umbralB ] = max( maxVARb );
```

Después de la generalización del algoritmo para el caso de tres umbrales, que también es posible de determinarlo con la función MULTITHRESH del software Matlab, se obtuvieron los siguientes umbrales por cada canal de color.

```
    >> k = 3;
    >> umbralR = multithresh(A1(:,:,1) ,k);
    >> umbralG = multithresh(A1(:,:,2) ,k);
    >> umbralB = multithresh(A1(:,:,3) ,k);
    >> size( umbralR )
```

Y luego se procedió a realizar la conversión de cada pixel de la imagen de color original a un valor cuantificado de 1 a 4, por corresponder a 4 clases diferentes. Esto fue ejecutado en cada una de los canales de la imagen de color.

```
    >> [f,c,a] = size(A1);
    >> for i=1:f
    for j=1:c
        if A1(i,j,1) <= umbralR(1), A1r(i,j) = 1; end
        if A1(i,j,1) > umbralR(1) & A1(i,j,1) <= umbralR(2), A1r(i,j) = 2; end
        if A1(i,j,1) > umbralR(2) & A1(i,j,1) <= umbralR(3), A1r(i,j) = 3; end
        if A1(i,j,1) >=umbralR(3), A1r(i,j) = 4; end
         if A1(i,j,2) <= umbralG(1), A1g(i,j) = 1; end
         if A1(i,j,2) > umbralG(1) & A1(i,j,2) <= umbralG(2), A1g(i,j) = 2; end
         if A1(i,j,2) > umbralG(2) & A1(i,j,2) <= umbralG(3), A1g(i,j) = 3; end
         if A1(i,j,2) >=umbralG(3), A1g(i,j) = 4; end
        if A1(i,j,3) <= umbralB(1), A1b(i,j) = 1; end
        if A1(i,j,3) > umbralB(1) & A1(i,j,3) <= umbralB(2), A1b(i,j) = 2; end
        if A1(i,j,3) > umbralB(2) & A1(i,j,3) <= umbralB(3), A1b(i,j) = 3; end
```

```
        if A1(i,j,3) >=umbralB(3), A1b(i,j) = 4; end
    end
    end
```

Seguidamente, en la figura 06 se representa el histograma de cada canal de color de la primera imagen original, con una señalización de los umbrales obtenidos por el método de Otsu.

```
>> subplot(131), imhist(A1(:,:,1));
>> hold on,
>> plot( [ umbralR(1) umbralR(1) ] , [ 0 1000 ] , 'r' );
>> plot( [ umbralR(2) umbralR(2) ] , [ 0 1000 ] , 'r' );
>> plot( [ umbralR(3) umbralR(3) ] , [ 0 1000 ] , 'r' );
>> subplot(132), imhist(A1(:,:,2));

>> hold on,
>> plot( [ umbralG(1) umbralG(1) ] , [ 0 1000 ] , 'g' );
>> plot( [ umbralG(2) umbralG(2) ] , [ 0 1000 ] , 'g' );
>> plot( [ umbralG(3) umbralG(3) ] , [ 0 1000 ] , 'g' );
>> subplot(133), imhist(A1(:,:,3));

>> hold on,
>> plot( [ umbralB(1) umbralB(1) ] , [ 0 1000 ] , 'b' );
>> plot( [ umbralB(2) umbralB(2) ] , [ 0 1000 ] , 'b' );
>> plot( [ umbralB(3) umbralB(3) ] , [ 0 1000 ] , 'b' );
>> hold off
```

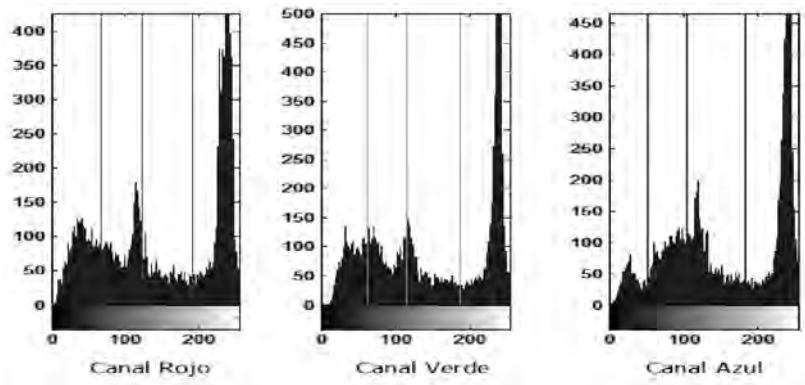

Figura 06. Señalización de los umbrales por cada canal de la imagen de color.

Operación de intersección lógica

Una vez realizada la umbralización de cada canal de color en cuatro clases diferentes, se procedió a realizar una operación de intersección basada en la función lógica AND, entre

la primera o mayor clase de la imagen de canal R y la última o menor clase de las imágenes de canal G y B. Por lo tanto, esto explica que si se encuentran pixeles muy cercanos a blanco en la imagen de canal R, así como pixeles próximos al color negro en las imágenes de los canales G y B, se tendrá un caso de pixel próximo al color rojo en la imagen original. De esta manera, el algoritmo de intersección está basado en un FOR anidado para recorrer todas las filas y columnas de las imágenes de interés. A continuación se muestra la expresión matemática, el procedimiento algorítmico realizado, y en la figura 07 el resultado de tal operación.

$$B = A1_r \geq 4 \quad \cap \quad A1_g \leq 2 \quad \cap \quad A1_b \leq 2 \tag{08}$$

```
>> [ f , c ] = size(A1);
>> for i=1:f
 for j=1:c
 if A1r(i,j) >= 4 & A1g(i,j) <= 2 & A1b(i,j) <= 2
 RA( i , j ) = 1;
 else
 RA( i , j ) = 0;
 end
 end
end
```

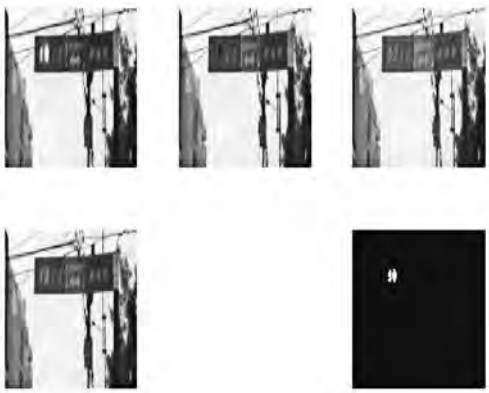

Figura 07. Resultado de la operación de intersección con la segunda imagen de color.

4. Operaciones morfológicas

A continuación se explica brevemente la definición correspondiente a la operación morfológica de erosión, así como de dilatación.

Operación morfológica de erosión

La morfología matemática se basa en operaciones de teoría de conjuntos. Particularmente en el caso de imágenes binarias, los conjuntos tratados son subconjuntos de Z^2, y en el de

las imágenes a tonos de gris, se trata de conjuntos de puntos en coordenadas en Z^3. Por lo cual, una operación morfológica de erosión, matemáticamente, es definida en términos de un grupo de operaciones entre dos conjuntos A y B en Z^2. Por lo cual, la erosión de A por B se define mediante la siguiente expresión matemática (Jain, 1989).

$$A \Theta B = \left\{ z \mid \left(B_z \cap A^C \neq \Phi \right) \right\} \tag{08}$$

Donde F representa al conjunto vacío, B al elemento estructural y A es la matriz correspondiente a la imagen a transformar. Un elemento estructural se entiende como una matriz o máscara que controlará la transformación que recibirá la imagen. Es decir, el incremento o decremento de pixeles blancos basado en una ley de referencia. Dicho elemento estructural puede estar formado por una matriz cuadrada de 3x3 o tomar cualquier otra forma particular que permita una satisfactoria transformación morfológica. Generalmente la erosión disminuye el tamaño de los objetos, dependiendo del elemento estructural utilizado. Una de las principales aplicaciones de la erosión es la eliminación de detalles irrelevantes de una imagen binaria. A continuación, la figura 08 muestra un resultado de aplicar la operación de erosión sobre la imagen resultante de la intersección lógica.

Para este trabajo se optó por utilizar un elemento estructural tipo cruz con la finalidad de disminuir la presencia de ruido en forma horizontal o vertical, que se encontrase en cualquier ubicación de la imagen.

$$B = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{09}$$

Operación morfológica de dilatación

Por el contrario, la operación morfológica de dilatación, en el caso de imágenes binarias, se encarga de aumentar los pixeles blancos también en función de la forma de un elemento estructural (Jain, 1989). Matemáticamente se representa de la siguiente manera:

$$A \oplus B = \left\{ z \mid \left(\hat{B}_z \cap A \neq \Phi \right) \right\} \tag{10}$$

Donde F representa al conjunto vacío, al elemento estructural y Bz es la matriz correspondiente a la imagen a transformar. Por lo general, la dilatación aumenta o dilata objetos en imágenes binarias, dependiendo del elemento estructural utilizado. Una de las principales aplicaciones de la dilatación es añadir pixeles al contorno de objetos presentes en una imagen. Asimismo, la figura 09 muestra un resultado de aplicar la operación de dilatación sobre la imagen erosionada, utilizando el mismo elemento estructural usado en la parte de la erosión.

```
>> B = [ 0 1 0 ; 1 1 1 ; 0 1 0 ];
>> EE = strel('arbitrary', B );
>> RAerosionado = imerode ( RA , EE );
>> figure(1)
>> colormap (gray(256))
>> subplot (121), image ( RA*255 ), axis off
>> subplot (122), image ( RAerosionado*255 ), axis off
```

Figura 08. Resultado de la operación de erosión.

```
>> EE = strel('arbitrary', B);
>> RAdilatado = imdilate( RAerosionado , EE );
>> figure(2)
>> colormap(gray(256))
>> subplot(121), image( RA*255 ) , axis off
>> subplot(122), image( RAdilatado*255 ) , axis off
```

Figura 09. Resultado de la operación de dilatación.

5. Reconocimiento de la luz roja

Después de la dilatación, tal como se observa en la figura 09, quedan dos grupos de pixeles de color blanco y donde uno de ellos es mayor que el otro. Esto indica que aparentemente, existen dos luces de color rojo en el semáforo analizado. Sin embargo, si observamos la fotografía superior derecha de la figura 01, el semáforo solamente presenta una luz roja. La otra corresponde al contador que aparece en el semáforo y que también es presentado de color rojo. Por lo tanto, después de analizar el volumen de pixeles blancos pertenecientes propiamente al color de luz de interés, se determinó una cantidad igual a 52 pixeles.

Fue así, que una vez procesada la imagen se continuó con el etiquetado y cálculo del número de luces rojas, o en su defecto número de contadores y/o luces de color rojo presentes en el semáforo.

```
>> [ C , L] = bwlabel( RAdilatado , 4 );
>> Contador = max( L );
```

Finalmente, se sobrepone en la imagen de color original, un marco amarillo sobre la luz roja del semáforo, permitiendo así la visualización del mismo. De igual manera, también resultaría importante si se adiciona un aviso del tipo sonoro que puede desarrollarse en forma básica utilizando el puerto de audio del propio Matlab. A continuación, se presenta el código de programa utilizado para representar el marco de color amarillo sobre dos imágenes capturadas y procesadas.

```
>> A1 = double(A1);
>> [ f , c , a ] = size( A2 );
>> [ f1 , c1 ] = size( RAdilatado );
>> RAdilatado = [ RAdilatado ; zeros( f-f1 , c ) ];
>> [x,y] = find( RAdilatado == 1 );
>> x = round( mean(x) );
>> y = round( mean(y) );
>> TotalU = round( sum(sum( RAdilatado )) / (Contador*2) ) - 4;
>> RAFinal = A2;
>> [a,b1] = max( RAdilatado(x ,: ) );
>> [a,b2] = max(fliplr( RAdilatado(x ,: ) ));
>> b1 = b1 - 6;
>> b2 = c - b2 + 6;
>> columnas = b1:b2;
>> filas = x-round(TotalU) : x+round(TotalU);

>> RAFinal( x-round(TotalU),columnas,1:2 ) = ones( 1, length(columnas) , 2 )*255;
>> RAFinal( filas , b1 , 1:2 ) = ones( length(filas) , 1 , 2 )*255;
>> RAFinal( filas , b2 , 1:2 ) = ones( length(filas) , 1 , 2 )*255;
>> RAFinal( x+round(TotalU),columnas,1:2 ) = ones( 1, length(columnas) , 2 )*255;

>> RAFinal( x-round(TotalU) , columnas , 3 ) = zeros( 1 , length(columnas) , 1 );
>> RAFinal( filas , y-round(TotalU ) , 3 ) = zeros( length(filas) , 1 , 1 );
>> RAFinal( filas , y+round(TotalU) , 3 ) = zeros( length(filas) , 1 , 1 );
>> RAFinal( x+round(TotalU) , columnas , 3 ) = zeros( 1 , length(columnas) , 1 );

>> figure(5), image( RAFinal ), axis off
```

Figura 10. Resultado de la identificación de la luz roja.

Resultados alcanzados

En este artículo, se procedió a analizar los resultados desde el punto de vista visual. Es decir, solo fue necesario inspeccionar en la fotografía la correcta detección de la o las luces rojas en el semáforo aéreo, a partir de la aparición de un marco de color amarillo.

Figura 11. Resultado de la no identificación de la luz roja.

De esta manera, la técnica planteada fue aplicada al grupo de las 20 fotografías capturadas de distintos puntos de la ciudad de Lima. Por lo cual, en 19 de ellas se obtuvo un reconocimiento total de las luces rojas sin importar la cantidad de ellas. Sin embargo, hubo una fotografía donde no fue posible el reconocimiento en su totalidad. A continuación, la figura 11 muestra la fotografía donde no fue posible el reconocimiento de la luz roja en el

semáforo. Esto quizás se deba a la poca variabilidad de intensidades de gris en la imagen del canal rojo. Por lo cual, sería necesario realizar un manejo de contraste para conseguir pixeles más claros y así facilitar la identificación.

A continuación se muestra una tabla que representa el resumen del resultado de la técnica propuesta para la identificación de la luz roja del semáforo. Por lo cual, el porcentaje de error alcanzado es del 95% por conseguir el reconocimiento de la luz roja en 19 fotografías de un total de 20.

Item	Fotografías	Número de luces rojas	Porcentaje de reconocimiento
1	Foto1.jpg, Foto2.jpg, Foto3.jpg, Foto4.jpg, Foto5. jpg, Foto6.jpg, Foto7.jpg, Foto8.jpg, Foto9.jpg.	01	100 %
2	Foto10.jpg.	01	0 %
3	Foto11.jpg, Foto12.jpg, Foto13.jpg, Foto14.jpg, Foto15.jpg, Foto16.jpg.	02	100 %
4	Foto17.jpg, Foto18.jpg, Foto19.jpg, Foto20.jpg.	03	100 %

Conclusiones

La técnica de procesamiento digital de imágenes basado en una umbralización múltiple, permitió un reconocimiento de la luz roja del semáforo en un 95%. Debido a que solamente en una imagen no fue posible alcanzar tal objetivo. Principalmente este hecho se debió a que la imagen presentaba un contraste pobre, lo que no permitió diferenciar los pixeles blancos correspondientes a la luz roja, del fondo de la fotografía. Por lo cual, se concluye que esta metodología planteada puede mejorarse si se realizan las pruebas para un número mayor de fotografías, considerando que la aparición de limitaciones pueden ser subsanadas según el error que presenten.

Asimismo, también es posible realizar más pruebas considerando el cálculo de un mayor número de umbrales en la imagen, lo que permitiría contar con una variedad de opciones para el reconocimiento final de la luz roja en el semáforo.

Impacto esperado

La expectativa de este artículo es contribuir con una técnica matemática de ayuda al conductor, al momento de reconocer la luz roja encendida en el semáforo. Por lo cual, esto permitiría a los interesados mejorar la técnica matemática planteada, para extenderlo y aplicarlo al caso de un video digital. De esta manera, se lograría una ayuda en tiempo real evitando confusiones, distracciones o reacciones lentas en lo que se refiere al tránsito en una ciudad muy congestionada. Y, mejor aún, si se llega a implementar en un hardware embebido, con un procesador DSP para lograr la independización del Matlab y la PC o Laptop.

Referencias bibliográficas

Alcaim A., Santos O. C. *Fundamentos do processamento de Sinais de voz e imagem*. Editora Interciéncia. Rio de Janeiro - Brasil. 2011.

Cortés J., Muriel A. y Mendoza J. "Comparación cualitativa y cuantitativa de las técnicas básicas de umbralización global basadas en histogramas para el procesamiento digital de imágenes". Artículo de *Scientia et Technica* Año XVI, No 49, Universidad Tecnológica de Pereira, Colombia. 2011.

Gonzalez R. C., Woods R. E. and Eddins S. L. *Digital Image Processing using Matlab*. Prentice Hall. NJ. 2004.

Image Processing Toolbox, Users Guide, Version 4. The Math Works, Inc. Natick, MA. 2003.

Jain A. K., "Fundamentals of Digital Image Processing". Editorial Prentice Hall. New Jersey. 1989.

Pajares G., De la Cruz J. *Visión por computador. Imágenes digitales y aplicaciones*. Editorial Alfaomega, 2001.

Parker J. R. *Algorithms for Image Processing and Computer Vision*. Editorial AlfaOmega. 1997.

Vieira N. H., Marques F. O. *Processamento digital de imagens*. Editora Brasport. Rio de Janeiro-Brasil. 1999.