TRANSFORMACIONES GEOMÉTRICAS EN IMAGENES

Pimentel, Carlos Daniel; carlosdpimenteld@gmail.com
Ingeniería Electrónica. Unidades Tecnológicas de Santander.

Resumen—El presente artículo muestra el desarrollo del cuarto laboratorio de la cátedra Procesamiento Digital de Imágenes, donde se ponen a prueba los métodos para realizar la transformación geométrica o el cambio espacial de las imágenes. Se muestran, primero, los resultados obtenidos al aplicar tres de las diferentes clases de interpolación que existen a una imagen específica. Se realizan las principales transformaciones geométricas; y por último, se utiliza la aplicación de Video y Procesamiento de Imagenes de Simulink.

Índice de Términos— Interpolación, transformación espacial, transformación geométrica.

I. INTRODUCCIÓN

Dentro de lo que se denomina procesamiento digital de imágenes se pueden encontrar una serie de técnicas especiales aplicables a una imagen digital que permiten distorsionarla, es decir, los valores de los pixeles pueden cambiar su posición. Algunos de este tipo de operaciones son el desplazamiento, la rotación y el escalamiento o distorsión. Las operaciones geométricas son muy utilizadas en la práctica, especialmente en las actuales y modernas interfaces graficas de usuario y video juegos; en otras palabras, no existe ninguna aplicación gráfica que no tenga capacidades de zoom para resaltar pequeños aspectos de las imágenes. En el área de la computación de imágenes es importante la aplicación de las técnicas de transformaciones geométricas para la representación de toda clase de imágenes junto con sus características como la textura; también la podemos aplicar en temas como imágenes en 3D o simplemente para la representación de entornos en tiempo real. Sin embargo para que el usuario obtenga los mejores resultados, sin importar lo moderna que puede ser su computadora, es necesario el uso de un tiempo de máquina considerable [1].

II. CONTENIDO DEL ARTÍCULO

A. *Interpolación de Imágenes*

La interpolación es el método por el cual se calculan los valores de una función en aquellos puntos donde no existen relaciones exactas entre el valor de la función y el punto que lo representa. En las transformaciones geométricas realizadas sobre imágenes se necesita utilizar el proceso de interpolación en la transformación $T(x,y)$ [1].

A continuación se mostraran los códigos utilizados para la realización de las diferentes interpolaciones antes expuestas, junto con sus resultados.

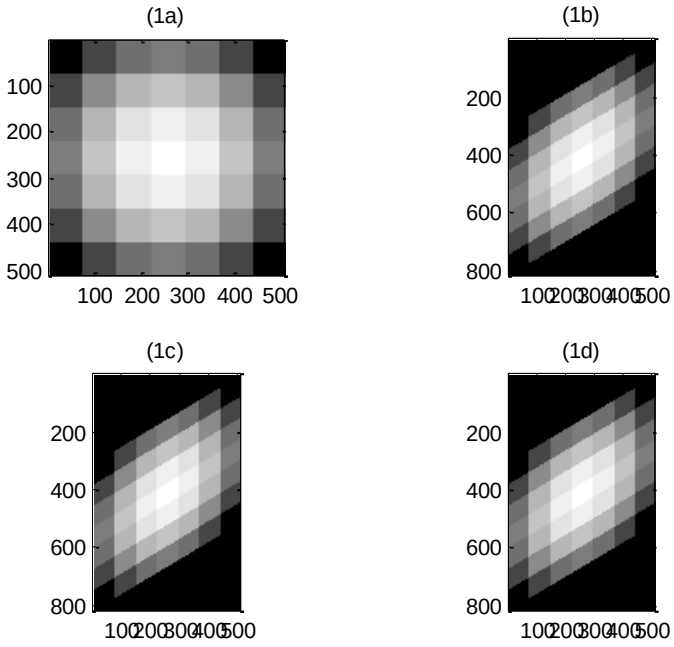

Fig. 1: Diferentes tipos de interpolación hechas a una imagen.

(1a) Imagen original en escala de grises; (1b) Interpolación por Vecino Próximo; (1c) Interpolación Bilineal; (1d) Interpolación Bicúbica.

En la siguiente figura (figura 2) se pueden apreciar los bordes de la misma imagen de la figura 1a aumentados de tamaño para poder diferenciar las técnicas aplicadas de interpolación.

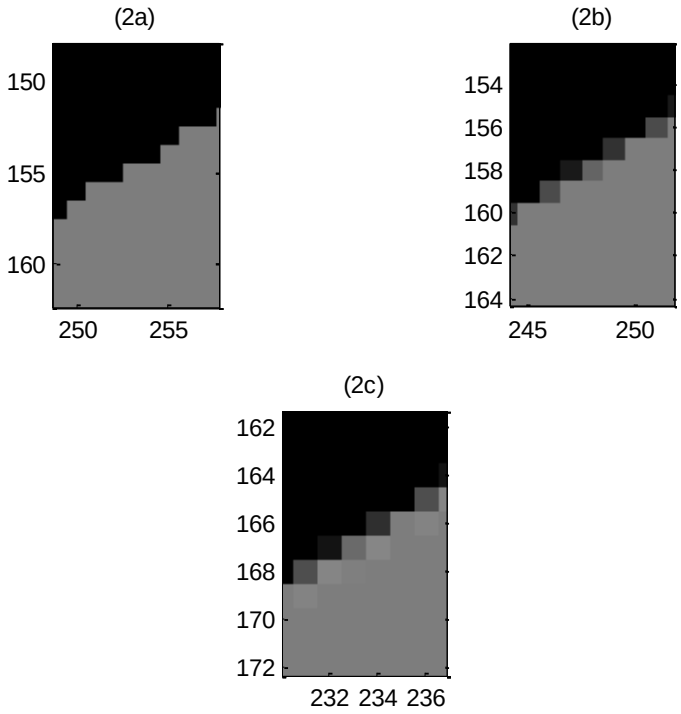

Fig. 2: Denotación de las diferentes clases de bordes obtenidos por las clases de interpolación. (2a) Interpolación por Vecino Próximo; (2b) Interpolación Bilineal; (2c) Interpolación Bicúbica.

El código usado fue el siguiente:

```
  I=imread('Contornos.jpg');
  II=rgb2gray(I);
  a=1; b=0; c=0; d=-.6; e=1; f=0;
  tform = maketform('affine',[a d 0; b e
0; c f 1]);
  % Tres transformaciones con diferente
interpolación
  IIIn = imtransform(II,tform,'nearest');
  IIIb = imtransform(II,tform,'bilinear');
  IIIbc = imtransform(II,tform,'bicubic');
  figure,subplot(2,2,1),subimage(II),Title
('(1a)');
  subplot(2,2,2),subimage(IIIn),Title('(1b
)');
  subplot(2,2,3),subimage(IIIb),Title('(1c
)');
  subplot(2,2,4),subimage(IIIbc),Title('(1
d)');
  figure,subplot(2,2,1),subimage(IIIn),Tit
le('(2a)');
  subplot(2,2,2),subimage(IIIb),Title('(2b
)');
  subplot(2,1,2),subimage(IIIbc),Title('(2
c)');
```

A continuación, podemos encontrar otro ejemplo en donde se aplican las diferentes clases de interpolación a la imagen seleccionada.

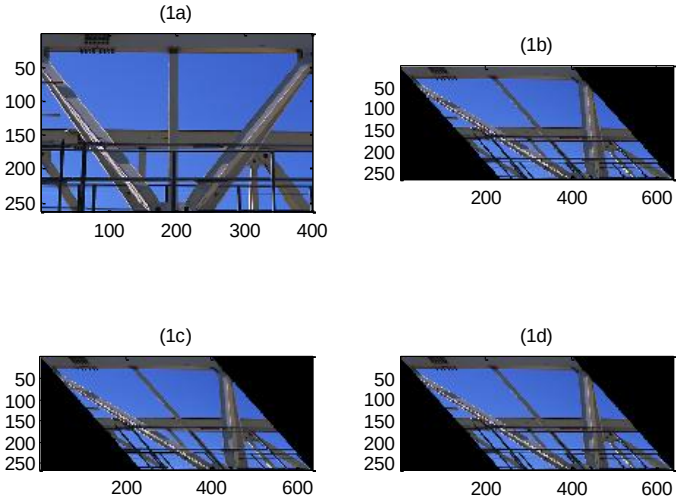

Fig. 3: Otro ejemplo de diferentes tipos de interpolación hechas a una imagen. (a) Imagen original en escala de grises; (b) Interpolación por Vecino Próximo; (c) Interpolación Bilineal; (c) Interpolación Bicúbica.

Como en el ejemplo anterior observaremos los bordes de la imagen aumentados para poder diferenciar los métodos.

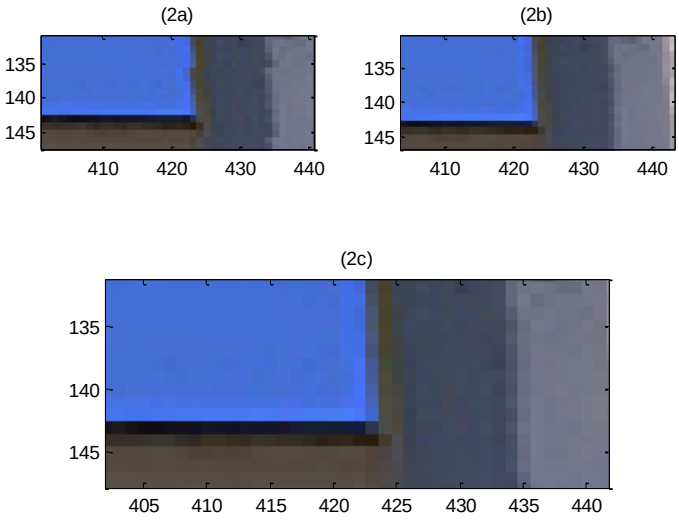

Fig. 4: Denotación de las diferentes clases de bordes obtenidos por las clases de interpolación de la imagen 1a de la figura 3. (2a) Interpolación por Vecino Próximo; (2b) Interpolación Bilineal; (2c) Interpolación Bicúbica.

B. Registro de Imagenes

A continuación procederemos a implementar, paso a paso, el código correspondiente para desarrollar la aplicación del registro de imágenes.

El código que usaremos será el siguiente:

```
  I=imread('face.jpg');
  II=rgb2gray(I);
  a=cos(t); b=-sin(t); c=0; d=sin(t);
e=cos(t); f=0;
  t1=[a d 0; b e 0; c f 1];
  tform = maketform('affine',t1);
  II45 = imtransform(II,tform);
  cpselect(II45, II);
```

Se abre le ventana que se muestra a continuación.

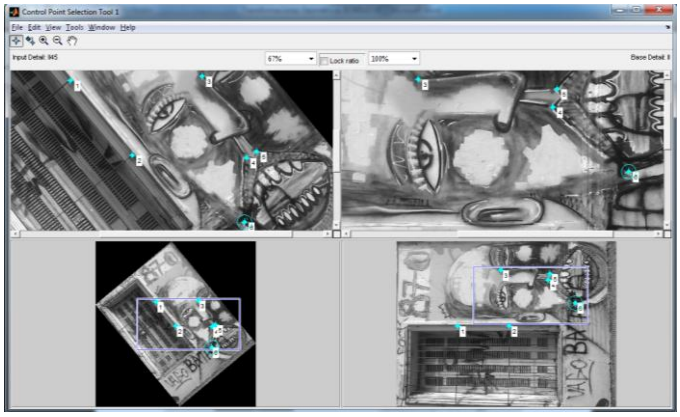

Fig. 5: Cuadro de la herramienta del Control del Punto de Selección.

Exportamos los puntos en File – Export Points to Workspace e insertamos el siguiente código.

```
  tform = cp2tform(input_points,
base_points, 'projective');
  [alineada xdata ydata] =
imtransform(II45, tform,'FillValues',
255);
  figure; imshow(alineada, 'XData', xdata,
'YData', ydata)
  hold on
  imshow(II);
```

El resultado final es el siguiente:

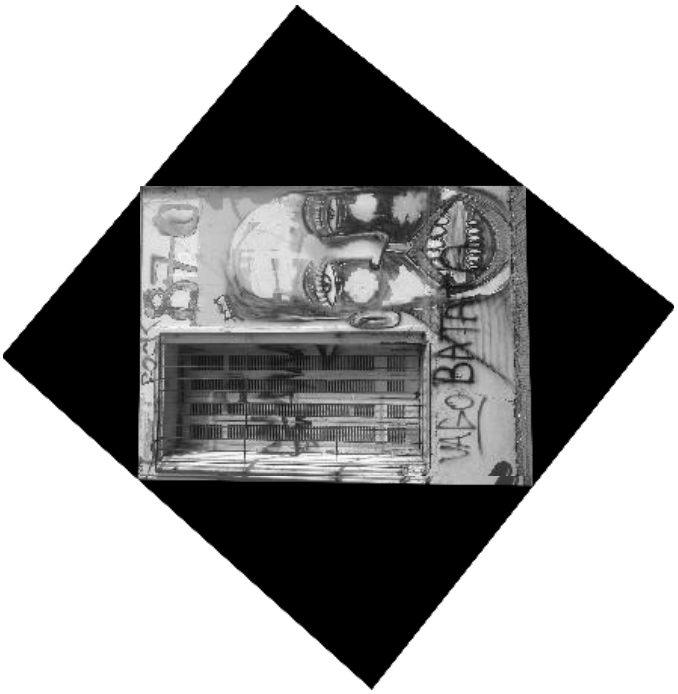

Fig. 6: Imagen terminada. Resultado de la alineación y posterior superposición de ambas imagenes.

C. Transformaciones Espaciales de las Imagenes

A las transformaciones espaciales también se les llama transformación afín. Este tipo de transformaciones se realiza mediante la utilización de coordenadas homogéneas, con las cuales se puede representar la combinación la combinación de las transformaciones traslación, escalamiento y rotación en la forma:

$$\begin{bmatrix} \hat{x}' \\ \hat{y}' \\ \hat{h}' \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} a_{12} a_{13} \\ a_{21} a_{22} a_{23} \\ 0 \quad 0 \quad 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (1)$$

Esta definición de transformación es conocida como transformación afín con seis grados de libertad a_{11},…, a_{23}, donde a_{11} y a_{23} definen la traslación, mientras que a_{11}, a_{12}, a_{21} y a_{22} definen el escalamiento, la inclinación y la rotación. A través de la utilización de la transformación espacial son transformadas líneas a líneas, triángulos a triángulos y rectángulos a paralelogramos. La

característica más importante en este tipo de transformación es que mantiene la relación existente entre las distancias de los puntos contenidos en las líneas de la imagen convertida [1].

Aplicando las diferentes transformaciones espaciales posibles a una imagen determinada se obtuvieron los siguientes resultados.

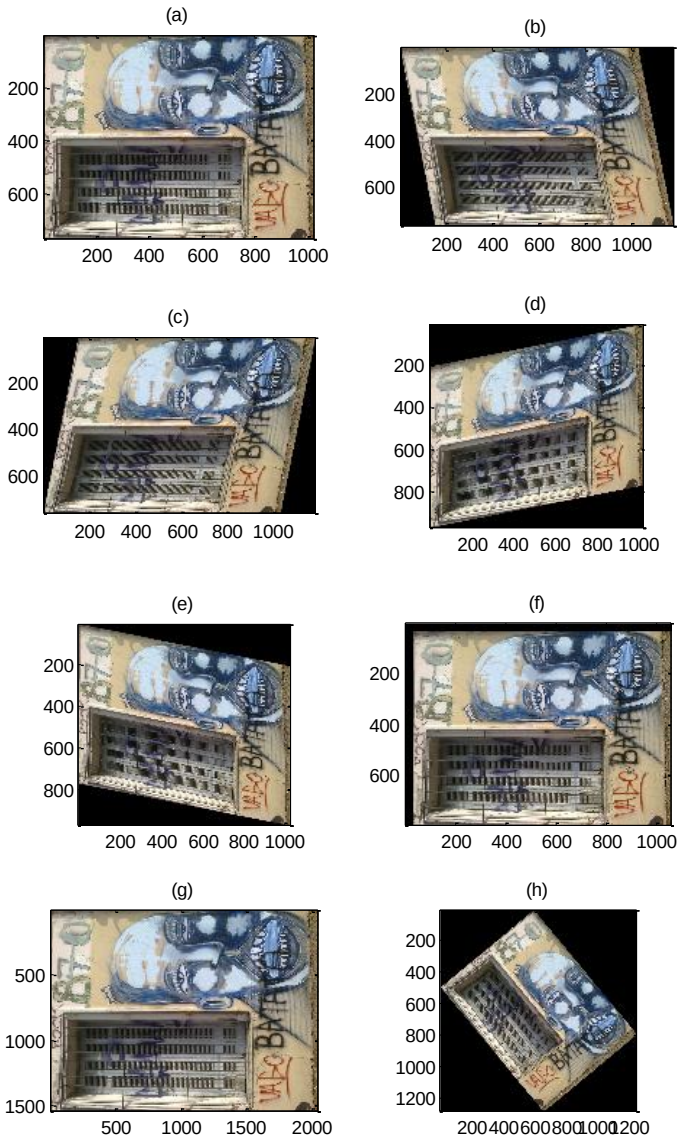

Fig. 7: Diferentes transformaciones geométricas aplicadas a una imagen específica. (a) Imagen original; (b) y (c) Empuje horizontal; (d) y (e) Empuje vertical; (f) Traslación; (g) Ampliación; (h) Rotación respecto a un ángulo.

El código usado fue el siguiente:

```
I = imread('face.jpg');
a=1; b=.2; c=0; d=0; e=1; f=0;
  tform = maketform('affine',[a d 0; b e
0; c f 1]);
  II = imtransform(I,tform);
  a=1; b=-.2; c=0; d=0; e=1; f=0;
  tform = maketform('affine',[a d 0; b e
0; c f 1]);
  III = imtransform(I,tform);
  a=1; b=0; c=0; d=-.2; e=1; f=0;
  tform = maketform('affine',[a d 0; b e
0; c f 1]);
  IV = imtransform(I,tform);
  a=1; b=0; c=0; d=.2; e=1; f=0;
  tform = maketform('affine',[a d 0; b e
0; c f 1]);
  V = imtransform(I,tform);
  a=1; b=0; c=30; d=0; e=1; f=30;
  xform=[a d 0; b e 0; c f 1];
  tform = maketform('affine',xform);
  VI = imtransform(I,tform,'XData', [1
(size(I,2)+ xform(3,1))],'YData', [1
(size(I,1)+ xform(3,2))]);
  a=5; b=0; c=0; d=0; e=5; f=0;
  tform = maketform('affine',[a d 0; b e
0; c f 1]);
  VII = imtransform(I,tform);
  t=45/(2*pi);
  a=cos(t); b=-sin(t); c=0; d=sin(t);
e=cos(t); f=0;
  tform = maketform('affine',[a d 0; b e
0; c f 1]);
  VIII = imtransform(I,tform);
  figure,
subplot(2,2,1),subimage(I),Title('(a)');
  subplot(2,2,2),subimage(II),Title('(b)')
;
  subplot(2,2,3),subimage(III),Title('(c)'
);
  subplot(2,2,4),subimage(IV),Title('(d)')
;
  figure,
subplot(2,2,1),subimage(V),Title('(e)');
  subplot(2,2,2),subimage(VI),Title('(f)')
;
  subplot(2,2,3),subimage(VII),Title('(g)'
);
  subplot(2,2,4),subimage(VIII),Title('(h)
');
```

D. Transformación Geométrica Compuesta

Es la transformación que se le aplica a una imagen, por medio una secuencia de transformaciones afines y cuyo fin es obtener una transformación geométrica total o compuesta.

A continuación procederemos a implementar,

paso a paso, el código correspondiente para desarrollar la aplicación de la transformación geométrica compuesta.

Los resultados fueron los siguientes:

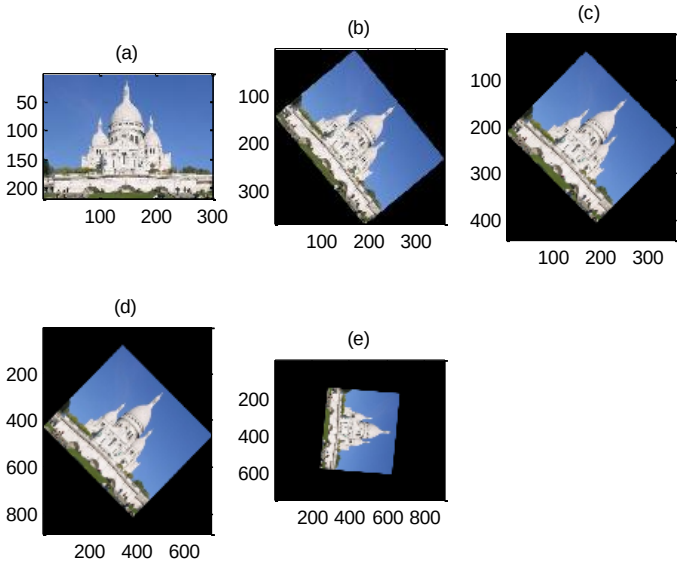

Fig. 8: Diferentes transformaciones geométricas compuestas aplicadas a una imagen especifica. (a) Imagen original; (b) Transformación con ángulo de rotación; (c) Empuje vertical; (d) Ampliación; (e) Producto matricial.

El código usado fue el siguiente:

```
  I=imread('iglesia.jpg');
  t=45/(2*pi);
  a=cos(t); b=-sin(t); c=0; d=sin(t);
e=cos(t); f=0;
  t1=[a d 0; b e 0; c f 1];
  tform = maketform('affine',t1);
  II = imtransform(I,tform);
  a=1; b=0; c=0; d=-.2; e=1; f=0;
  t2=[a d 0; b e 0; c f 1];
  tform = maketform('affine',t2);
  III = imtransform(II,tform);
  a=5; b=0; c=0; d=0; e=5; f=0;
  t3=[a d 0; b e 0; c f 1];
  tform = maketform('affine',t3);
  IV = imtransform(III,tform);
  tr=t3*t2*t1;
  tform = maketform('affine',tr);
  V = imtransform(II,tform);
  figure,
subplot(2,3,1),subimage(I),Title('(a)');
  subplot(2,3,2),subimage(II),Title('(b)')
;
  subplot(2,3,3),subimage(III),Title('(c)'
);
  subplot(2,3,4),subimage(IV),Title('(d)')
;
  subplot(2,3,5),subimage(V),Title('(e)');
```

E. Algoritmo de Kmeans Clustering

A continuación procederemos a implementar, paso a paso, el código correspondiente para desarrollar la aplicación del algoritmo de Kmeans Clustering.

```
  I=imread('face0.jpg');
  II=rgb2gray(I); II=im2bw(II);
  I90=imread('face90.jpg');
  II90=rgb2gray(I90); II90=im2bw(II90);
  IIc=imcomplement(II);
  [filas,columnas]=find(IIc==1);
  I3=[filas,columnas];
  [U, v]=kmeans(I3,3); xy=round(v);
  II90c=imcomplement(II90);
  [filas,columnas]=find(II90c==0);
  I4=[filas,columnas];
  [U, v]=kmeans(I4,3);
  xy90=round(v);
  tform = maketform('affine',xy,xy90);
  [x,y] = tforminv(tform, xy90(:,1),
xy90(:,2));
  [u,v] = tformfwd(tform,x,y);
  figure; imshow(II); hold on;
  plot(xy(:,1),xy(:,2),'o',x,y,'*');
```

El resultado fue el siguiente:

Fig. 9: Imagen a 0° con sus tres puntos significativos.

En la figura 9 se pueden apreciar los puntos significativos que el código usado encontró en la imagen a 0° y que están señalados por flechas rojas.

Ahora aplicamos el siguiente código:

```
figure; imshow(II90); hold on;
plot(xy90(:,1),xy90(:,2),'o',u,v,'*');
```

El resultado fue el siguiente:

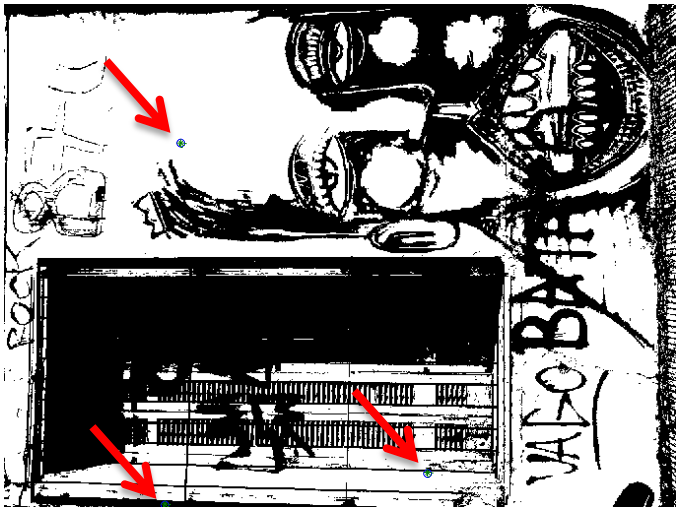

Fig. 10: Imagen a 90° con sus tres puntos significativos.

En la figura 10 se pueden apreciar los puntos significativos que el código usado encontró en la imagen a 90° y que están señalados por flechas rojas.

Por último, aplicamos el siguiente código:

```
  xy90p=[xy90(1,:) 1; xy90(2,:) 1;
xy90(3,:) 1];
  xyp=[xy(1,:) 1;xy(2,:) 1;xy(3,:) 1];
  T=xyp\xy90p;
  tform = maketform('affine',T);
  II90r=imtransform(II,tform);
  figure; imshow(II90r)
```

El resultado final es el siguiente:

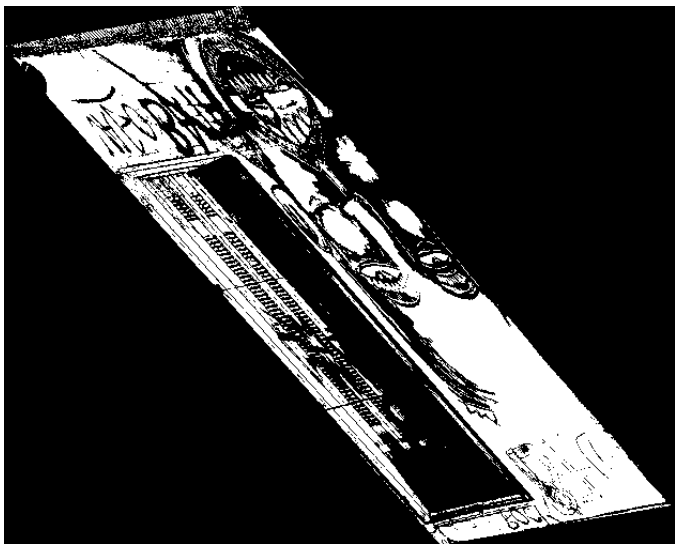

Fig. 11: Imagen resultante aplicando el algoritmo de Kmeans Clustering.

F. Aplicación del Video and Image Processing Blockset de Simulink

El diagrama de bloques implementado en Simulink fue el siguiente:

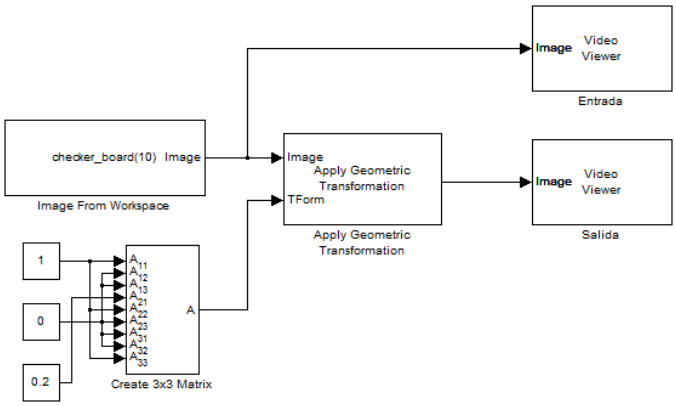

Fig.: 12: Diagrama de bloques utilizado en Simulink para aplicación de la transformación geométrica.

La imagen usada fue la siguiente:

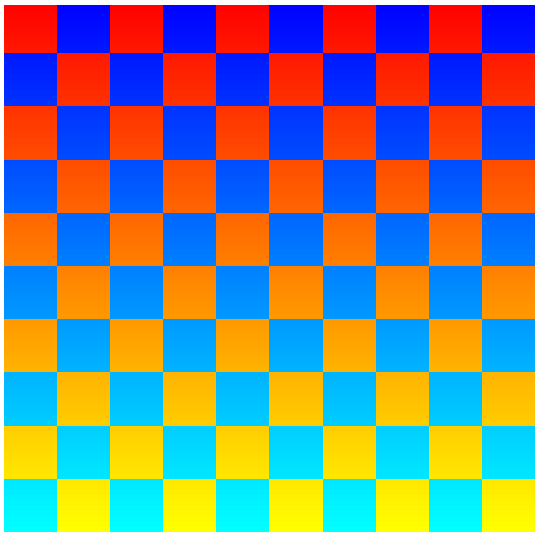

Fig. 13: Imagen de entrada llamada *Checker_board(10)* que utiliza MATLAB por defecto en Simulink.

Una vez simulado la transformación geométrica se obtuvo el siguiente resultado:

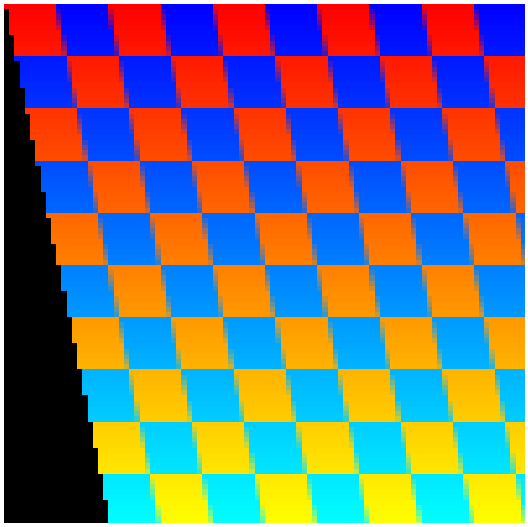

Fig. 14: Imagen de salida resultado de la simulación en Simulink.

III. CONCLUSIONES

Las diferentes clases de interpolación permiten al usuario obtener variados y mejores resultados cuando a bordes se refiere, según los criterios y la exigencia que se desee.

Cuando se trabaja con las tres diferentes clases de interpolación se puede observar que la técnica de vecinos cercanos produce bordes fuertes y para nada suaves; mientras que, las técnicas bilineal y bicúbica proporcionan bordes suaves y delicados.

La técnica de transformación bicúbica proporciona bordes mucho más delicados permitiéndole a cada objeto que se encuentra en esta, complementarse con los demás objetos de la imagen, que la técnica bicúbica.

La técnica de transformación geométrica o afín no altera los puntos que pertenecen a la imagen original manteniendo las distancias entre estos y dejando la imagen final igual en su forma y tamaño a la original.

Para poder realizar la interpolación, la transformación geométrica o afín y la transformación espacial se utiliza, al igual que los filtros lineales, una matriz máscara o kernel, que en este caso se llama matriz de transformación, y les permite crear, por medio de la función *'maketform'*, una estructura de transformación, llamada TFORM

con la que posteriormente se hará la trasformación deseada [1].

REFERENCIAS

[1] C. Erik, Z. Daniel, P. Marco. Procesamiento Digital de Imágenes con MATLAB y Simulink. Primera Edición. México D.F.: Editorial Alfa Omega, 2011.
--, Operaciones Geométricas en Imágenes. México D.F., 2011. 639 p – 702 p.
--, Transformación de coordenadas. México D.F., 2011. 641 p – 649 p.
--, Interpolación. México D.F., 2011. 671 p – 682 p.
--, Funciones para la Transformación Geométrica en MATLAB. México D.F., 2011. 682 p – 687 p.