Practica 5: Interpolación

Aguilera Palacios Luis Ernesto, leapvader.1998@hotmail.com
Padilla Castillo Aaron Samir, samir.castill@gmail.com,
Facultad de Ingeniería, Universidad Nacional Autónoma de México

***Resumen*— En este trabajo se utilizarán varios tipos de interpolación para evaluar su desempeño y comparar su coste computacional.**

***Abstract*— In this work, various types of interpolation will be used to evaluate their performance and compare their computational cost.**

Objetivos

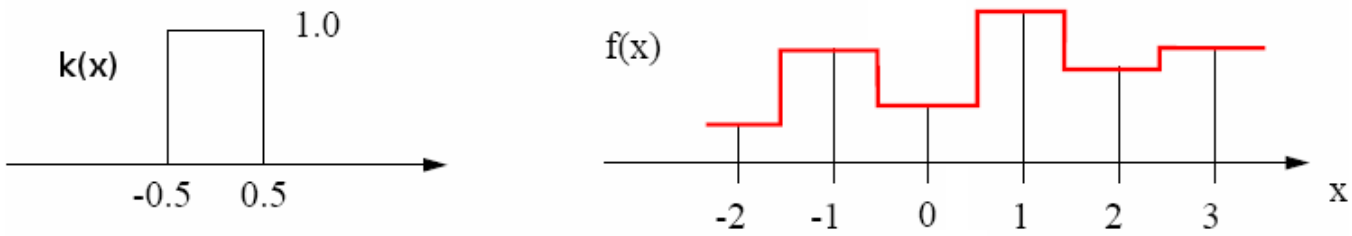
Dada una imagen con un número de muestras igual a N×M, donde N y M son el ancho y el largo de la imagen respectivamente:

- Encontrar las imágenes interpoladas usando interpoladores de orden cero, lineal y cúbico.
- Interpolar la imagen en el dominio de la frecuencia.

Introducción

Interpolación de vecino más cercano

"En el método de interpolación del vecino más cercano, el valor de píxel interpolado es el de su vecino más cercano. El índice del vecino más cercano se encuentra mediante la operación de redondeo. Al redondear un número decimal, el número se reemplaza por el siguiente entero mayor si la parte fraccionaria es 0.5 o mayor. De lo contrario, se reemplaza por el número entero más grande no mayor que él mismo." [3]

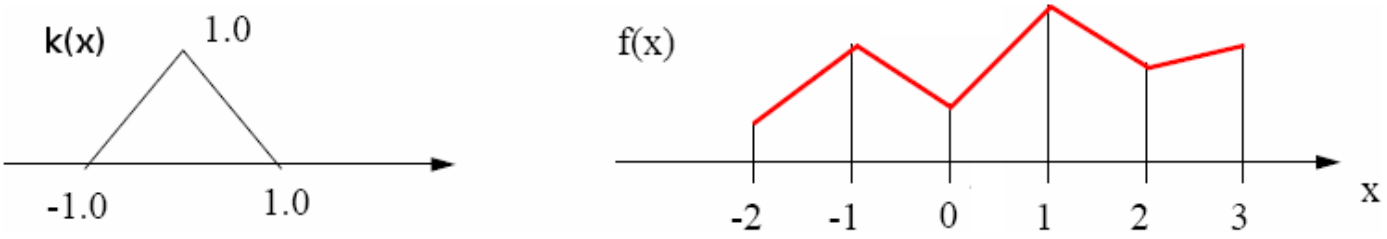


Fuente: https://clouard.users.greyc.fr/Pantheon/experiments/rescaling/index-en.html.

Interpolación bilineal

"En la interpolación lineal, se supone que se traza una línea recta entre dos puntos vecinos y el valor interpolado es una combinación lineal de las distancias de los vecinos y sus valores. En la interpolación bilineal, la interpolación lineal se realiza en cada una de las dos coordenadas ortogonales de la imagen… Este método se usa a menudo en la práctica y la imagen interpolada es menos borrosa en comparación con el algoritmo del vecino más cercano. La interpolación está determinando los valores de una función dentro del rango de valores conocidos. Por lo tanto, obtenemos una salida de 7 × 7 para una entrada de 4 × 4. Si necesitamos una salida de 8 × 8, se requiere una extrapolación. Se pueden utilizar extensiones de borde adecuadas (o métodos de extrapolación), como el que se utiliza para las operaciones de vecindad." [3]

*



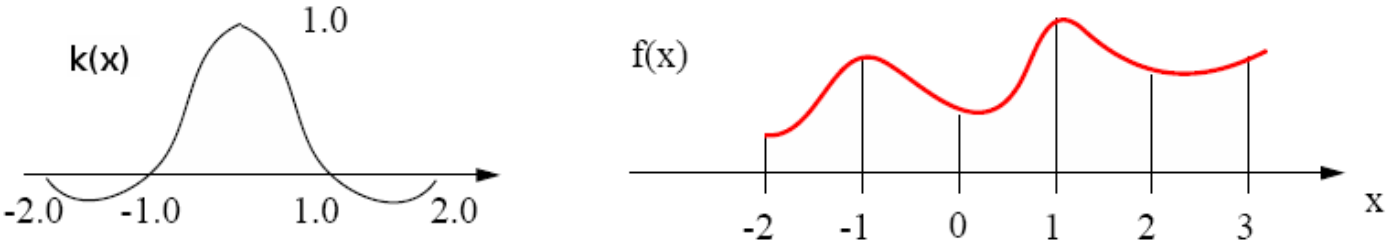
Fuente: https://clouard.users.greyc.fr/Pantheon/experiments/rescaling/index-en.html.

Interpolación bicúbica

"El núcleo de interpolación de convolución cúbica se compone de polinomios cúbicos por partes definidos en los subintervalos (- 2, - l), (- 1, 0), (0, l) y (1, 2). Fuera del intervalo (- 2, 2), el núcleo de interpolación es cero. Como consecuencia de esta condición, el número de muestras de datos utilizadas para evaluar la función de interpolación en se reduce a cuatro. El núcleo de interpolación debe ser simétrico. Junto con la condición anterior, esto significa que debe tener la ecuación:

$$u(s) = \begin{cases} A_1|s|^3 + B_1|s|^2 + C_1|s| + D_1 & ; \quad 0<|s|<1 \\ A_2|s|^3 + B_2|s|^2 + C_2|s| + D_2 & ; \quad 1<|s|<2 \\ 0 & ; \quad 2<|s| \end{cases}$$

" [1]



Fuente: https://clouard.users.greyc.fr/Pantheon/experiments/rescaling/index-en.html.

I. Desarrollo

Para todos los puntos siguientes y con la finalidad de poder observar el desempeño de los distintos interpoladores usar una imagen nítida de baja resolución. Se recomienda usar imágenes desde 128x128 hasta 256x256 pixeles como máximo y la imagen *pentagon* que se encuentra disponible en la sección de imágenes del curso.

Figura 1. Imagen original.

1. Sobremuestreo espacial:

- Obtenga el sobremuestreo de la imagen original insertando ceros entre los pixeles de la misma con factores T ↑= 2×2 y T ↑= 4×4.

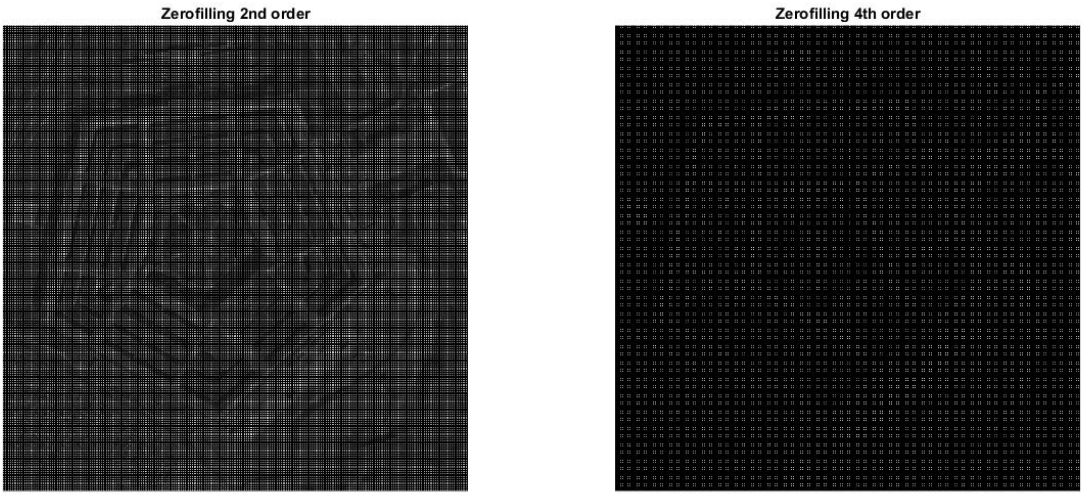


Figura 2. Imágenes con sobremuestreo sin calcular (*zerofilling*).

Se pueden observar los huecos que están con ceros.

- Obtenga la magnitud del espectro de la DFT (abs) de cada una de las imágenes sobremuestreadas y de la imagen original. Despliegue los resultados en una misma figura para efectos de comparación (en Matlab se puede usar el comando subplot). Recuerde usar fftshift para centrar los espectros y una función de escalamiento para el despliegue, ejemplo: ImFDespliegue=log(1.0 + ImF), donde ImF es la DFT de la imagen con sobremuestreo.

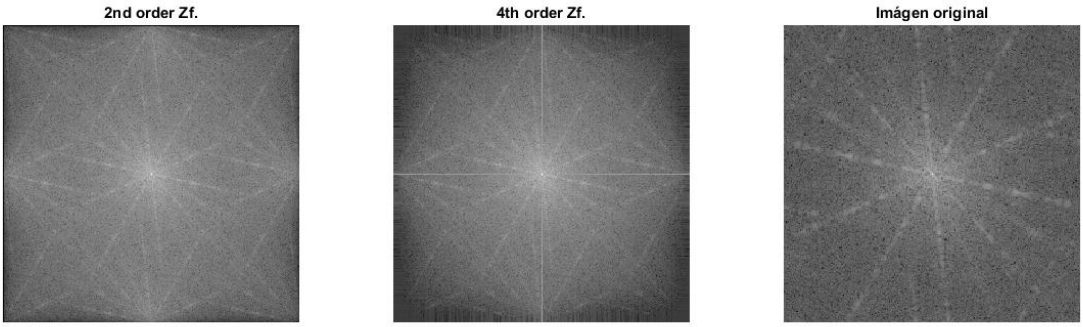


Figura 3. Transformadas de Fourier de los sobremuestreos sin calcular.

Podemos observar como en los bordes comienzan a aparecer oscuros.

- Explique el efecto del sobremuestreo espacial en el espectro de la DFT. ¿Qué sucedería si se intercalaran ceros entre los valores del espectro de la DFT?

Se puede apreciar que en los contornos se pierde información (frecuencias altas) y que en general la imagen comienza a distinguirse cada vez menos. Si se intercalan ceros en la transformada creo que la pérdida de información sería mucho mayor en las imágenes originales.

2. Interpolación espacial. Interpole las imágenes con sobremuestreo obtenidas en el inciso anterior (con factores $T\uparrow = 2\times2$ y $T\uparrow = 4\times4$) usando interpoladores:

   i. De orden cero
   ii. Lineal
   iii. Cúbico

- Para cada caso y factor de interpolación despliegue con alguna función de acercamiento (zoom) una misma región seleccionada de las imágenes interpoladas. Compare y explique los resultados de los distintos interpoladores en una misma figura.

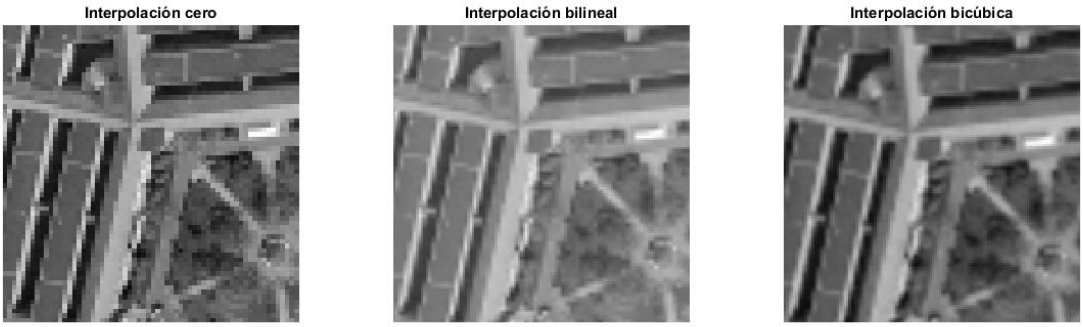


Figura 4. Interpolaciones de orden 2.

Figura 5. Interpolación de orden 4.

Aunque la interpolación bicúbica nos da una interpolación muy buena nos consume una cantidad muy grande de tiempo a comparación de la bilineal o la cero.

En la última imagen podemos apreciar con bastante detalle como las figuras pueden cambiar con interpolaciones distintas, por ejemplo la interpolación cero vemos figuras con líneas cortadas, en la bilineal podemos apreciar mejor las figuras pero el contraste se pierde un poco y por lo tanto los árboles no se aprecian correctamente, y en la bicúbica podemos ver como nos da mejores contrastes e incluso podemos ver los árboles.

- Obtenga la magnitud del espectro de la DFT (abs) de cada una de las imágenes interpoladas. Despliegue los espectros en una misma figura para compararlos. Explique el efecto de los distintos tipos de interpolación en los espectros de las DFTs.

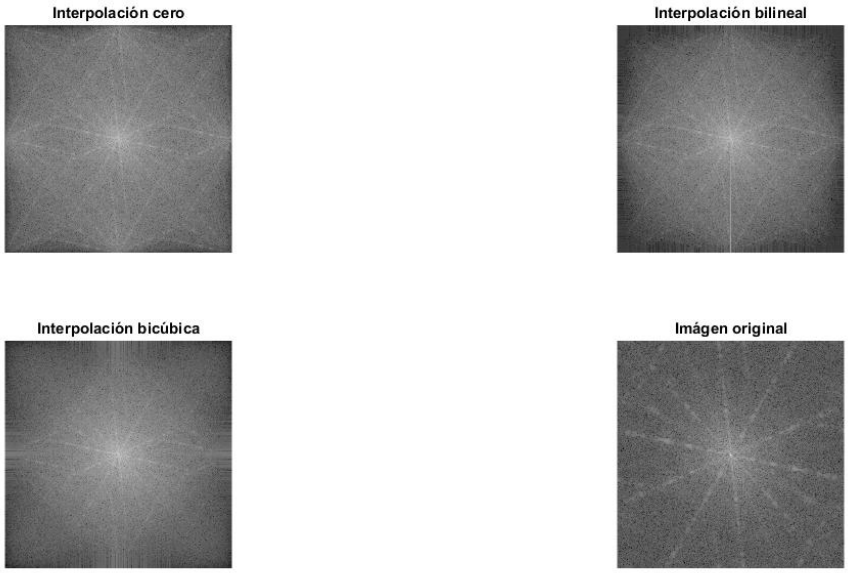


Figura 6. Interpolación de orden 2.

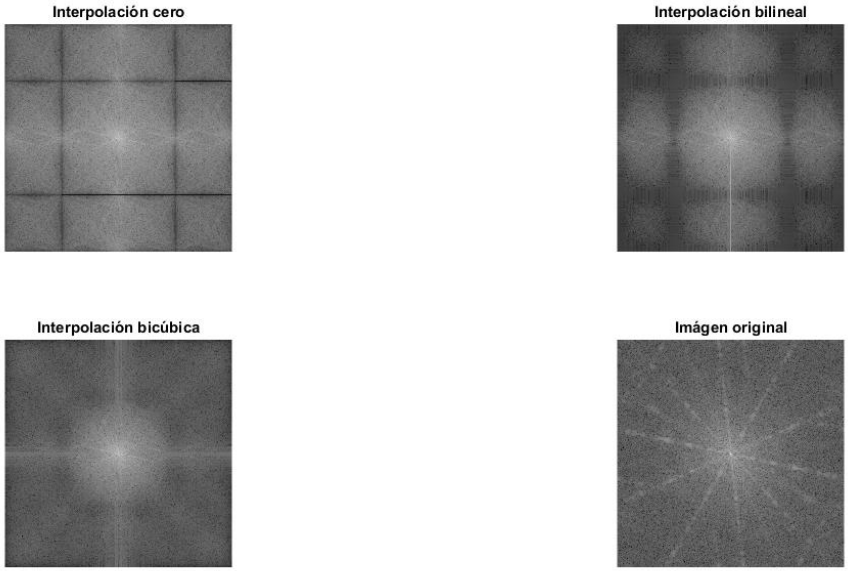


Figura 7. Interpolación de orden 4.

Aquí podemos obsevar como la interpolación bicúbica nos permite obtener resultados bastante buenos(perdemos poca información) a costa de un coste computacional alto.

3. Interpolación en frecuencia:

- Obtenga la DFT de la imagen original y despliegue su magnitud.

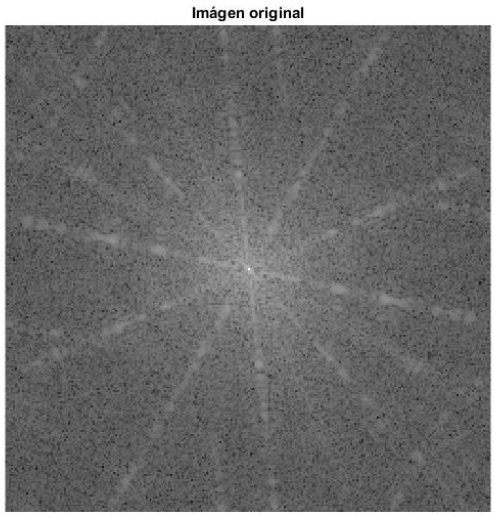


Figura 8. DFT de la imagen original.

- Centre el espectro de la DFT (fftshift) y agregue ceros alrededor del mismo hasta completar y obtener dos espectros cuyas dimensiones correspondan a las dimensiones de la imagen original multiplicadas por los factores de interpolación T ↑= 2×2 y T ↑= 4×4 respectivamente.

- Despliegue las magnitudes de las DFTs (abs) con ceros alrededor. Use una función de escalamiento para el despliegue.

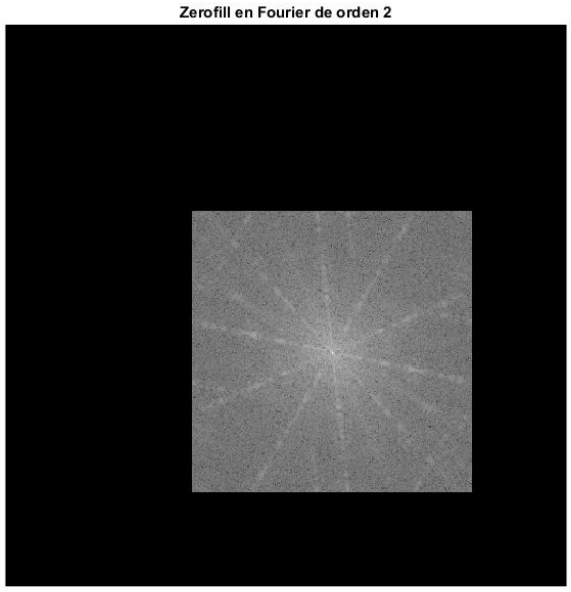


Figura 9. Llenado de ceros en el dominio de Fourier de orden 2.

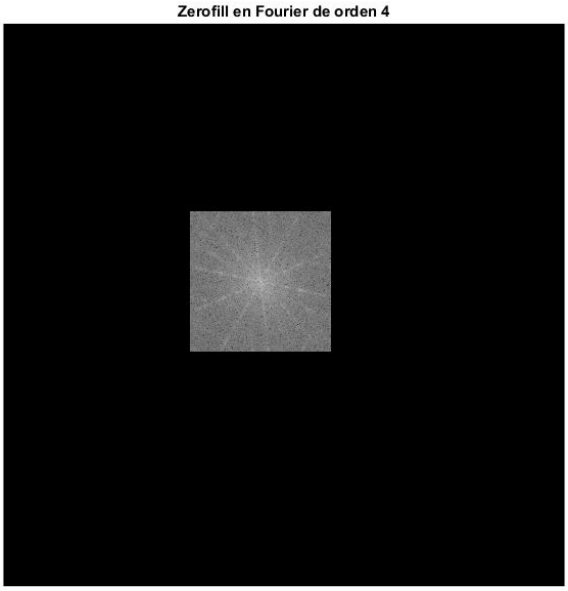


Figura 10. Llenado de ceros en el dominio de Fourier de orden 4.

- Compare en una misma figura las magnitudes de las DFTs (abs) de las imágenes interpoladas en el punto 2 con la magnitudes de las DFTs (abs) con ceros alrededor. Explique los resultados y diferencias.

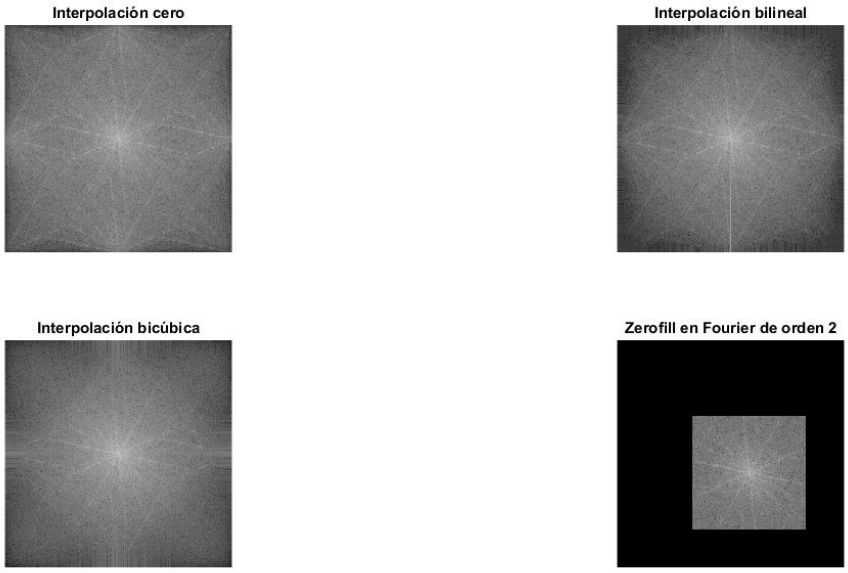


Figura 11. Magnitudes de las DFT de orden 2 comparadas.

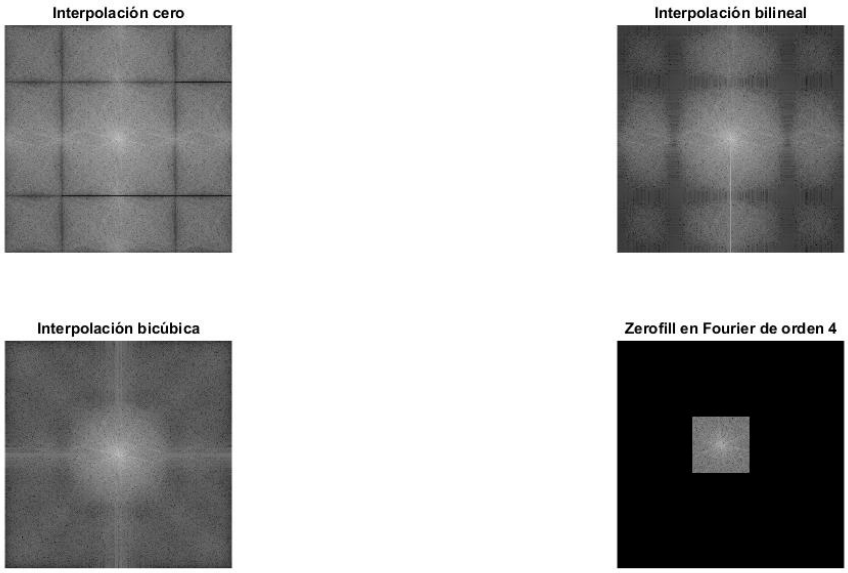


Figura 12. Magnitudes de las DFT de orden 4.

- Calcule la inversa de la DFT (IDFT) de las DFTs con ceros alrededor.

Figura 13. Imagen de la transformada inversa en orden 2 después de rellenar con ceros.

Figura 14. Imagen de la transformada inversa en orden 4 después de rellenar con ceros.

Podemos observar como en Fourier el sobremuestreo no nos pone un coste computacional tan alto como con el bicúbico obteniendo aún así buenos resultados.

4. Compare en una misma figura los resultados de las imágenes interpoladas que se obtuvieron con los cuatro métodos.

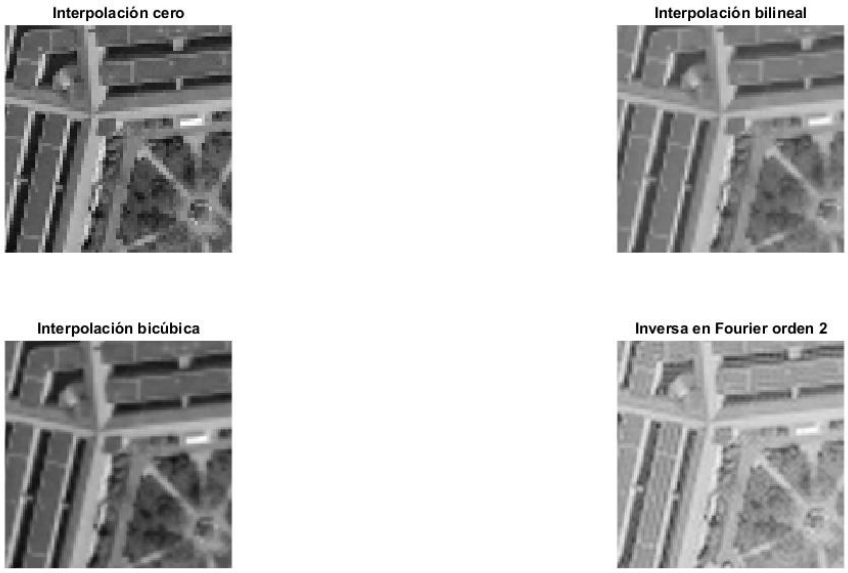


Figura 15. Comparación de las interpolaciones en orden 2.

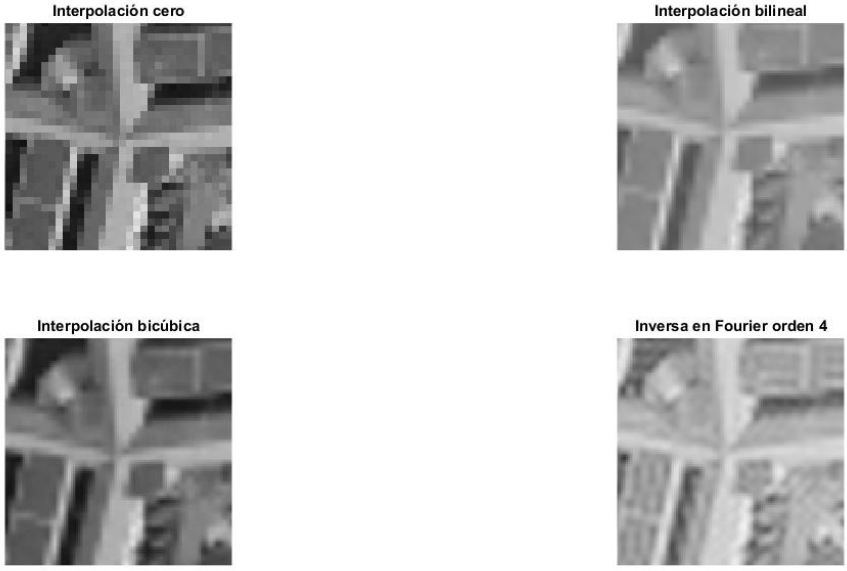


Figura 16. Comparación de las interpolaciones en orden 4.

II. Código

```
%Aguilera Ernesto
%Padilla Aaron

img = imread('pentagon256x256.tif', 'tif');

%%% Zerofilling de las imágenes
n=2;
i2=zero_fill_interp(img, n);
subplot(1,2,1);
imshow(i2, []);
title('Zerofilling 2nd order');
n=4;
i4=zero_fill_interp(img, n);
subplot(1,2,2);
imshow(i4, []);
title('Zerofilling 4th order');
mat_c4 =cubic_interp(img, n);
subplot(1,3,3);
imshow(mat_c4, []);
title('Interpolación bicúbica');

%%% Transformadas de Fourier del inciso anterior
fz4 = log(abs(1.0+fftshift(fft2(mat_z4))));
subplot(2,2,1);
imshow(fz4, []);
title('Interpolación cero');
fl4 = log(abs(1.0+fftshift(fft2(mat_l4))));
subplot(2,2,2);
imshow(fl4, []);
title('Interpolación bilineal');
fc4 = log(abs(1.0+fftshift(fft2(mat_c4))));
subplot(2,2,3);
imshow(fc4, []);
title('Interpolación bicúbica');
fo = log(abs(1.0+fftshift(fft2(img))));
subplot(2,2,4);
imshow(fo, []);
title('Imágen original');
```

```
fc2 = log(abs(1.0+fftshift(fft2(mat_c2))));
subplot(2,2,3);
imshow(fc2, []);
title('Interpolación bicúbica');
fo = log(abs(1.0+fftshift(fft2(img))));
subplot(2,2,4);
imshow(fo, []);
title('Imágen original');

%%% Sobremuestreo espacial de orden 4
n=4;

mat_z4 = zero_interp(img, n);
subplot(1,3,1);
imshow(mat_z4, []);
title('Interpolación cero');

mat_l4 = linear_interp(img, n);
subplot(1,3,2);
imshow(mat_l4, []);
title('Interpolación bilineal');
mat_l2 = linear_interp(img, n);
subplot(1,3,2);
imshow(mat_l2, []);
title('Interpolación bilineal');

mat_c2 =cubic_interp(img, n);
subplot(1,3,3);
imshow(mat_c2, []);
title('Interpolación bicúbica');

%%% transformadas de Fourier del inciso anterior
fz2 = log(abs(1.0+fftshift(fft2(mat_z2))));
subplot(2,2,1);
imshow(fz2, []);
title('Interpolación cero');
fl2 = log(abs(1.0+fftshift(fft2(mat_l2))));
subplot(2,2,2);
imshow(fl2, []);
title('Interpolación bilineal');
```

```
%%% Transformadas de Fourier del inciso anterior
f2 = log(abs(1.0+fftshift(fft2(i2))));
subplot(1,3,1);
imshow(fz2, []);
title('2nd order Zf.');
f4 = log(abs(1.0+fftshift(fft2(i4))));
subplot(1,3,2);
imshow(fl2, []);
title('4th order Zf.');
fo = log(abs(1.0+fftshift(fft2(img))));
subplot(1,3,3);
imshow(fo, []);
title('Imágen original');

%%% sobremuestreo espacial de orden 2
n=2;

mat_z2 = zero_interp(img, n);
subplot(1,3,1);
imshow(mat_z2, []);
title('Interpolación cero');
%%%Interpolación en frecuencia
fo = log(abs(1.0+fftshift(fft2(img))));
figure(2);
imshow(fo, []);
title('Imágen original');

n=2;
ff2 = zero_fill_fourier(fo, n);
figure(2);
imshow(ff2, []);
title('Zerofill en Fourier de orden 2');
n=4;
ff4 = zero_fill_fourier(fo, n);
figure(2);
imshow(ff4, []);
title('Zerofill en Fourier de orden 4');
%%%comparaciones
subplot(2,2,1);
imshow(fz2, []);
title('Interpolación cero');
subplot(2,2,2);
imshow(fl2, []);
title('Interpolación bilineal');
subplot(2,2,3);
imshow(fc2, []);
title('Interpolación bicúbica');
subplot(2,2,4);
imshow(ff2, []);
title('Zerofill en Fourier de orden 2');
```

```
%%%transformadas inversas
subplot(2,2,1);
imshow(mat_z2, []);
title('Interpolación cero');
subplot(2,2,2);
imshow(mat_l2, []);
title('Interpolación bilineal');
subplot(2,2,3);
imshow(mat_c2, []);
title('Interpolación bicúbica');
subplot(2,2,4);
imshow(I_frec_t2_final, []);
title('Inversa en Fourier orden 2');
```

Fig. A. Programa principal.

```
function [r] = cubic_interp(I, ratio)
    [h, w] = size(I);
    fh = (ratio * h);
    fw = (ratio * w);
    r = zeros(fh,fw);
    hs = (h/fh);
    ws = (w/fw);
    for i=1:fh
      y = (hs * i) + (0.5 * (1 - 1/ratio));
        for j=1:fw
            x = (ws * j) + (0.5 * (1 - 1/ratio));
      %// Any values out of acceptable range
          x(x < 1) = 1;
          x(x > h - 0.001) = h - 0.001;
          x1 = floor(x);
          x2 = x1 + 1;
          y(y < 1) = 1;
          y(y > w - 0.001) = w - 0.001;
          y1 = floor(y);
          y2 = y1 + 1;
       %// 4 Neighboring Pixels
          NP1 = I(y1,x1);
          NP2 = I(y1,x2);
          NP3 = I(y2,x1);
          NP4 = I(y2,x2);
       %// 4 Pixels Weights
          PW1 = (y2-y)*(x2-x);
          PW2 = (y2-y)*(x-x1);
          PW3 = (x2-x)*(y-y1);
          PW4 = (y-y1)*(x-x1);
          r(i,j) = PW1 * NP1 + PW2 * NP2 + PW3 * NP3 + PW4 * NP4;
        end
    end
  end
```

Fig. B. Interpolación bicúbica

```
function [r] = zero_interp(mat, n)
    kernel = ones(1, n+1);
    kernel(n+1)=0;
    ra = zero_fill_interp(mat, n);
    numRows = size(ra, 1);
    r = zeros(size(ra, 1)+n, size(ra, 2)+n);
    for i=1:numRows
       if mod(i, n)==1
            aux = conv(kernel, ra(i, :));
       end
       r(i,:)=aux;
    end
    r=r(1:size(ra, 1),1:size(ra,2));
end
```

Fig. C. Interpolación cero.

```
%Zero filling for the interpolation
function [r] = zero_fill_interp(mat, order)
    [numRows,numCols] = size(mat);
    b = order-1;
    r = zeros(numRows*order, numCols*order);
    for i=1:numRows
        for j=1:numCols
            r(order*i-b, order*j-b) = mat(i, j);
        end
    end
end
```

Fig. D. Llenado de ceros en espacial.

```
function [r] = linear_interp(mat, n)
    kernel = ones(1, 2*n-1);
    x = 1;
    for i=1:size(kernel,2)
        kernel(i) = x;
        if i<n
            x = x+1;
        elseif i >= n
            x = x-1;
        end
    end
    kernel = kernel/n;

    mat_z = conv2(kernel.', kernel, zero_fill_interp(mat, n));
    r = mat_z(n:size(mat_z, 1)-n+1,n:size(mat_z, 2)-n+1);
end
```

Fig. E. Interpolación bilineal.

```
%%%
I_frec = fft2(img);
temp = fftshift(I_frec);
%Padding de ceros, con la dft en el centro
I_frec_t2 = complex(zeros(size(img,2)*2));
I_frec_t2(size(img,2)/2+1:size(img,2)+size(img,2)/2,size(img,2)/2+1:size(img,2)+size(img,2)/2) = temp;
%I_frec_t2(129:384,129:384) = temp; es lo mismo de arriba.
I_frec_t2_final = log(abs(ifft2(I_frec_t2)));
imshow(I_frec_t2_final, []);
title('Inversa en Fourier orden 2');

%Padding de ceros, con la dft en el centro
I_frec_t4 = complex(zeros(size(img,2)*4));
%I_frec_t2(size(img,2)/2+1:size(img,2)+size(img,2)/2,size(img,2)/2+1:size(img,2)+size(img,2)/2) = temp,
I_frec_t4(341:596,341:596) = temp;
I_frec_t4_final = log(abs(ifft2(I_frec_t4)));
imshow(I_frec_t4_final, []);
title('Inversa en Fourier orden 4');
```

Fig. F. Llenado de zeros para Fourier.

III. Conclusiones

Al hacer interpolaciones de imágenes es importante saber que características queremos conservar para poder elegir un método de interpolación adecuado, pues el costo computacional aumenta rápidamente al hacer interpolaciones con mayores puntos y con curvas más suaves, por ejemplo en una aplicación para IA donde se procesen millones de imágenes es inviable usar un sobremuestreo con cúbicas por el tiempo que consume, sin embargo una interpolación cero podría ayudarnos si lo que buscamos es suficientemente grande y no requiere de algunos detalles que se pierden al sobremuestrear la imagen. En el dominio de Fourier obtenemos buenas aproximaciones con un coste computacional reducido, por lo que parece ser el que mejor relación costo/beneficio tiene.

Referencias

[1]
R. Keys, «Cubic Convolution Interpolation for Digital Image Processing », *IEEE Transactions on acoustics, speech and signal processing*, vol. ASSP-29, p. 8, dic. 1981.

[2]
R. Clouard, «Tutorial: Image Rescaling», *The Pantheon project*, 18-may-2011. [Online]. Disponible en: https://clouard.users.greyc.fr/Pantheon/experiments/rescaling/index-en.html. [Accedido: 06-dic-2020]

[3]
D. Sundararajan, Digital Image Processing. Singapore: Springer Singapore, 2017 [Online]. Disponible en: http://link.springer.com/10.1007/978-981-10-6113-4. [Accedido: 06-dic-2020]