PROYECTO de DITHERING

Apellidos, Nombre: Flores Flores, Daniel Javier
Apellidos, Nombre: Lillo Ramirez, Lucas
Apellidos, Nombre:

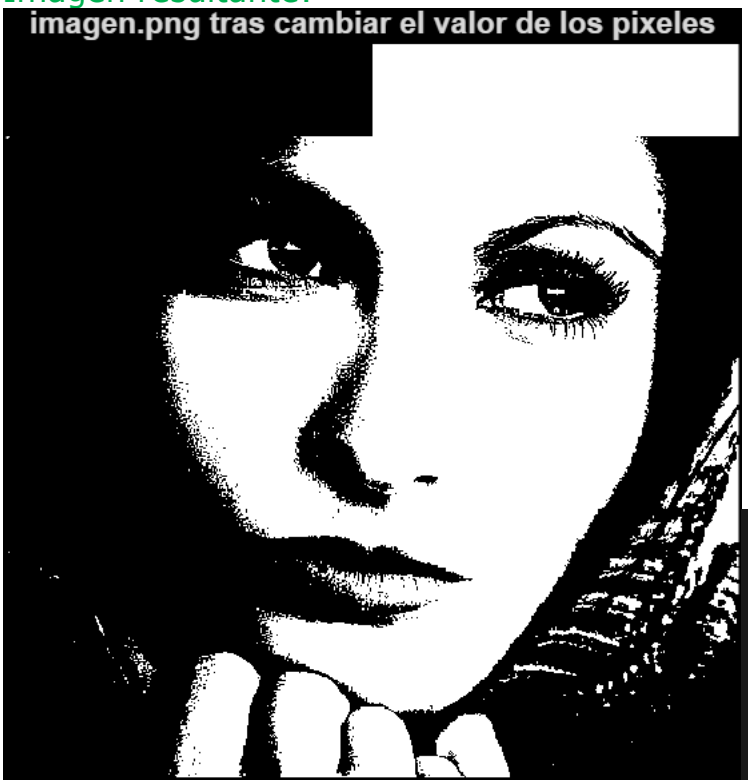
La entrega definitiva del proyecto será el 23 de febrero. La entrega será este fichero con las respuestas/código e imágenes pedidas.

Para la clase del 16 de febrero debéis llevar hecho hasta el punto indicado. En esa clase revisaré/puntuaré lo que tengáis hecho, resolveremos dudas, seguiréis trabajando con el resto de los apartados, ...

En esta práctica, al copiar las imágenes de dithering tratad de mantened su tamaño en el documento. Si se cambia pueden aparecer efectos raros al eliminar o recolocarse los puntos.

Adjuntad la imagen resultante y la desviación standard de la resta entre la imagen original y el resultado.

Imagen resultante:



Desviación standard obtenida:

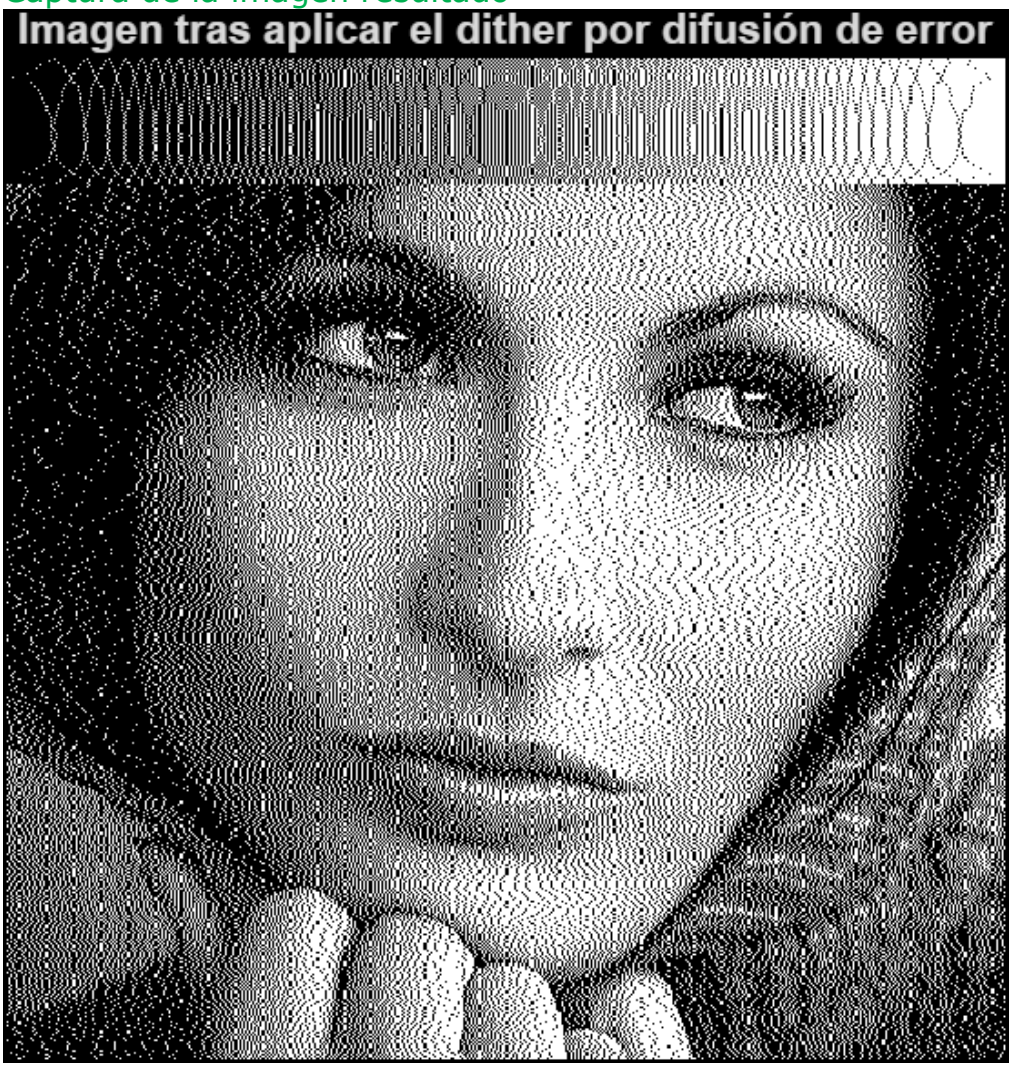
```
>> lab_1

desviacion_standard =

    0.2839
```

1. Dithering por difusión del error

Adjuntad código de **`function im=dither(im)`** y una captura de la imagen resultado.
El código es el siguiente:

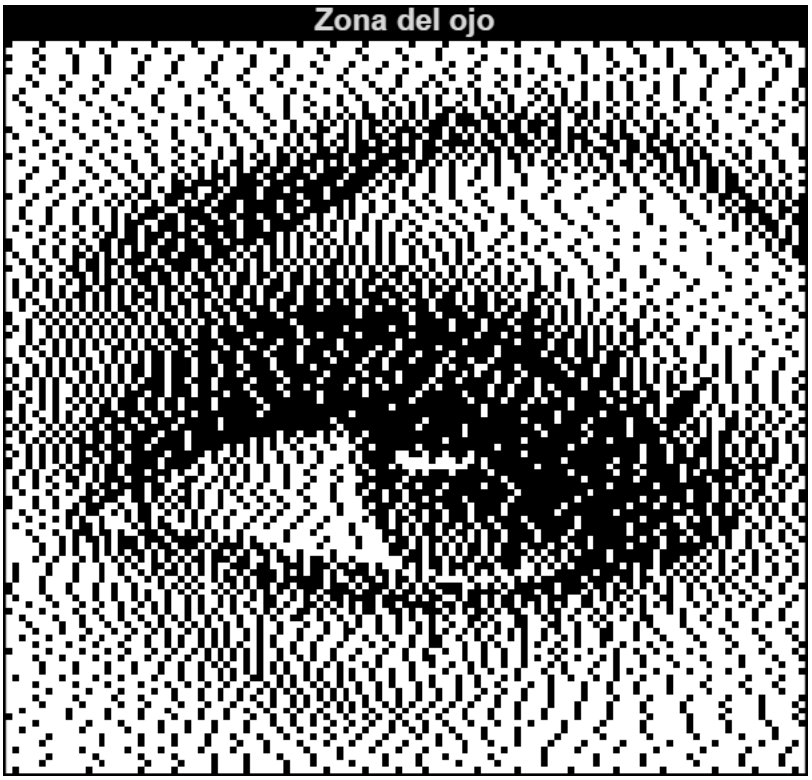
```
% ===== DITHERING POR DIFUSION DE ERROR =====
function im = dither(im)
    % Convertimos a valores entre 0 y 1 (im2double)
    im = im2double(im);
    % inicializamos error
    e=0;
    % Recorremos la imagen pixel a pixel
    [filas, columnas] = size(im);
    for fila = 1:filas
        % si la fila es impar, recorremos de izquierda a derecha
        if (mod(fila,2)) == 1
            for columna = 1:columnas
                % Obtenemos el valor del pixel actual sumando el error
                V = im(fila, columna) + e;
                % Comparamos V con 0.5
                if V >= 0.5
                    im(fila, columna) = 1;
                else
                    im(fila, columna) = 0;
                end
                  % Recalculamos el error cometido para poder usarlo con el siguiente
pixel
                e = V - im(fila, columna);
            end
        else
        % si la fila es par, recorremos de derecha a izquiera
            for columna = columnas:-1:1
                % Obtenemos el valor del pixel actual sumando el error
                V = im(fila, columna) + e;
                % Comparamos V con 0.5
                if V >= 0.5
                    im(fila, columna) = 1;
                else
                    im(fila, columna) = 0;
                end
                  % Recalculamos el error cometido para poder usarlo con el siguiente
pixel
                e = V - im(fila, columna);
            end
        end
    end
end % end de la funión
```

Captura de la imagen resultado



Adjuntad captura de la zona del ojo.

Captura de la zona del ojo:



¿Cuál es el principal problema que se aprecia visualmente en la imagen resultante?
El principal problema diríamos que es que en la imagen resultante se pueden ver unas líneas verticales, que se marcan más en las zonas de grises.
¿Cuál es su causa?
Al aplicar el método sobre la zona de los grises, dónde los valores están alrededor del 0.5. Un pixel que vale 0.5 o casi 0.5 el resultado es 0, al acumular el error el siguiente pixel (al que le sumamos este error) valdrá 1, por lo que los valores van oscilando y se crea ese efecto visual
Volcad su resultado de σ . ¿Mejora respecto al método anterior?

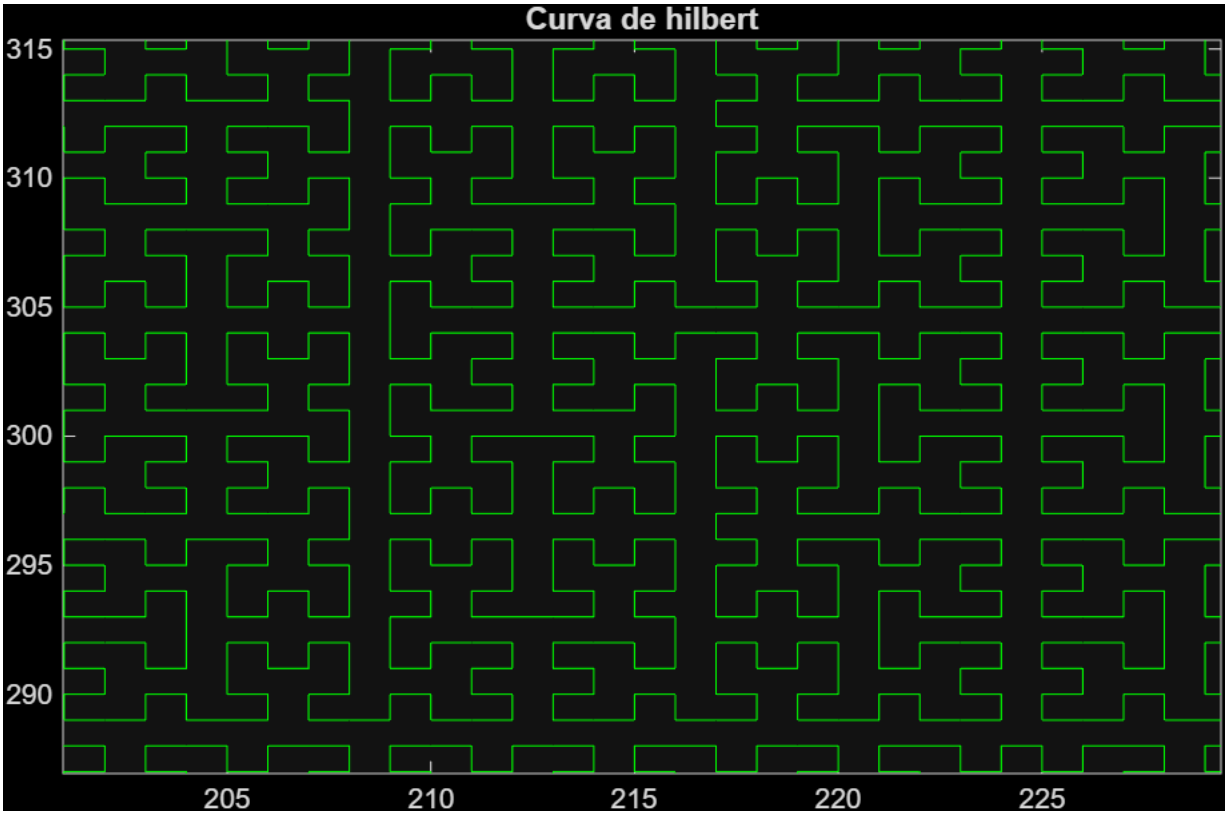
```
81
82  % Volvemos a calcular la desviación estandar de la diferencia entre la original y el resultado que hemos obtenido
83  desviacion_standard_1 = std2(im2double(imread('FC_PROY1/imagen.png'))- resultado)
84
```

```
>> lab_1

desviacion_standard_1 =

    0.4058
```

El resultado no mejora respecto al método del principio, de hecho empeora

¿Es adecuado este criterio matemático para juzgar la calidad "visual" del resultado?
Diríamos que no es adecuado, ya que a pesar de obtener un peor error con el método dithering la calidad visual que obtenemos con este método es mucho mejor y más cercana a la imagen original

Adjuntad captura del zoom de la curva de Hilbert

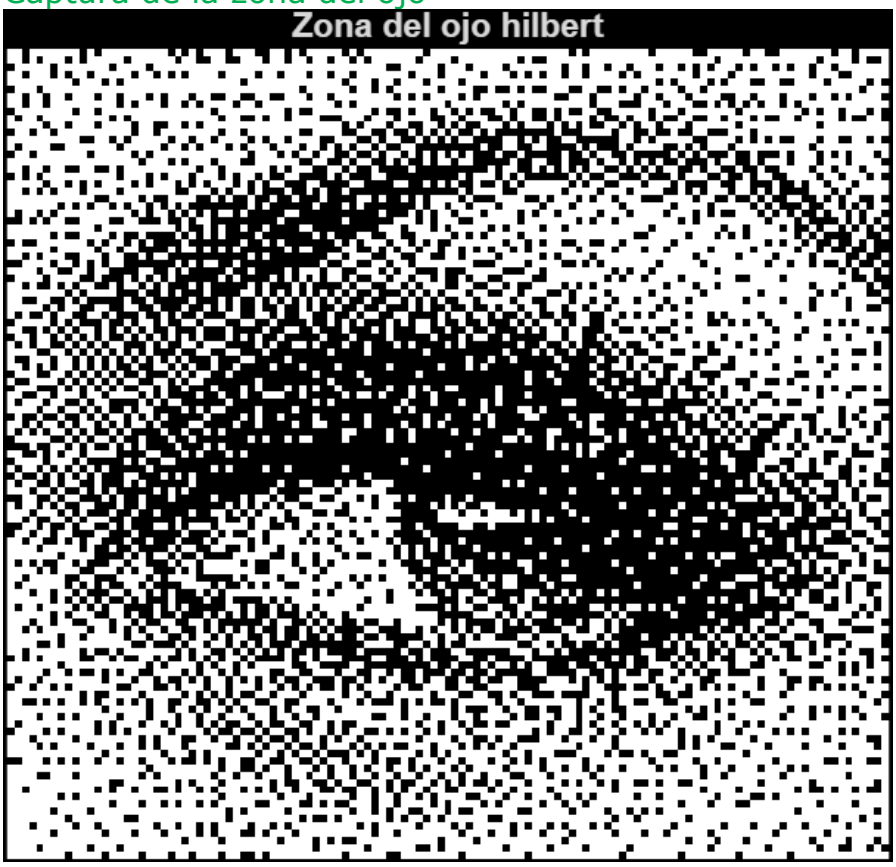


Adjuntad el código de **function** **im=dither_hilbert(im)** y una captura de la imagen resultado junto con el detalle como antes del ojo derecho.

```
% Nueva función utilizando la curva de Hilbert
function im = dither_hilbert(im)
    % Cargamos curva de Hilbert
    load('FC_PROY1/hilbert.mat');
    % Convertimos a valores entre 0 y 1
    im = im2double(im);
    % Inicializamos e error
    e = 0;
    % Recorremos la imagen
    for k = 1:length(I)
        % Obtenemos coordenadas
        fila = I(k);
        columna = J(k);
        % Obtener el valor del píxel actual sumando el error
        V = im(fila, columna) + e;
        % Comparar V
        if V >= 0.5
            im(fila, columna) = 1;
        else
            im(fila, columna) = 0;
        end
        % Recalcular el error cometido para usarlo en el siguiente píxel
        e = V - im(fila, columna);
    end
end
```

Captura del resultado con la curva de hilbert

Captura de la zona del ojo



Adjuntad código de la nueva función y una captura del resultado aplicado a la imagen test junto con el detalle del ojo.

Código de la función:

```
% funcion para repatir el error a sus 3 vecinos
function im = dither_3vecinos(im)
    % Convertimos a valores entre 0 y 1 (im2double)
    im = im2double(im);

    % Recorremos la imagen pixel a pixel
    [filas, columnas] = size(im);

    for fila = 1:filas
        % si la fila es impar recorremos de izquierda a derecha
        if (mod(fila,2)) == 1
            for columna = 1:columnas
                % Obtenemos el valor del pixel actual
                V = im(fila, columna);

                % Comparamos V con 0.5
                if V >= 0.5
                    im(fila, columna) = 1;
                else
                    im(fila, columna) = 0;
                end
```

```
        % Calculamos el error cometido para usarlo con el siguiente
        e = V - im(fila, columna);

        % Difundimos el error a 3 vecinos
        % Derecha (4/8)
        if columna+1 <= columnas
            im(fila, columna+1) = im(fila, columna+1) + e*(4/8);
        end
        % Abajo (3/8)
        if fila+1 <= filas
            im(fila+1, columna) = im(fila+1, columna) + e*(3/8);
        end
        % Abajo-derecha (1/8)
        if (fila+1 <= filas) && (columna+1 <= columnas)
            im(fila+1, columna+1) = im(fila+1, columna+1) + e*(1/8);
        end
    end
else
    % si la fila es par, recorremos de derecha a izquierda
    for columna = columnas:-1:1
        % Obtenemos el valor del pixel actual
        V = im(fila, columna);

        % Comparamos V con 0.5
        if V >= 0.5
            im(fila, columna) = 1;
        else
            im(fila, columna) = 0;
        end

        % Calculamos el error cometido
        e = V - im(fila, columna);

        % Difundimos el error a 3 vecinos
        % Izquierda (4/8)
        if columna-1 >= 1
            im(fila, columna-1) = im(fila, columna-1) + e*(4/8);
        end
        % Abajo (3/8)
        if fila+1 <= filas
            im(fila+1, columna) = im(fila+1, columna) + e*(3/8);
        end
        % Abajo-izquierda (1/8)
        if (fila+1 <= filas) && (columna-1 >= 1)
            im(fila+1, columna-1) = im(fila+1, columna-1) + e*(1/8);
        end
    end
end
```

```
    end
end % end de la funion
```

Captura de la zona del ojo:

Haced hasta este punto previo a la clase de LAB siguiente.

Resul	Comp directa	Prop 1 filas	Prop 1 hilbert	Prop 3 pix
σ(res-I_org)	0.2839	0.4058	0.4058	0.4035

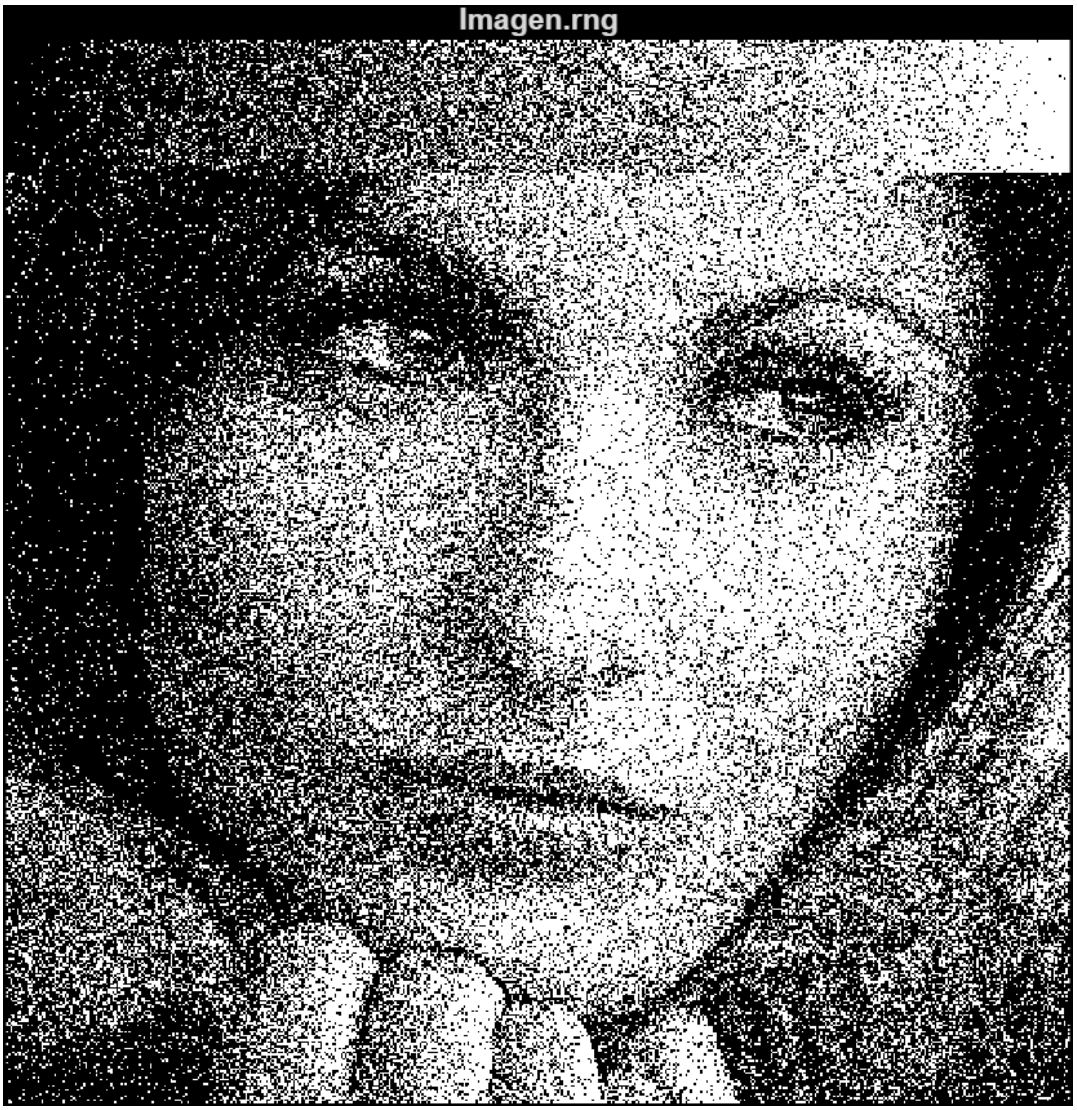
Resul	Comp directa	Prop 1 filas	Prop 1 hilbert	Prop 3 pix
σ(res_filt-I_org)	0.2522	0.0490	0.0480	0.0442

2. "Ordered dithering"

Adjuntad el código usado (se valorará no usar bucles) y la imagen resultado. Dad el valor de la desviación standard de la diferencia entre el original y el resultado filtrado como se hizo en el apartado anterior.

```
% ===== Ordered dithering =====

im = imread('FC_PROY1/imagen.png');
im = im2double(im);
im2 = im;
```

```
%Creamos la imagen random
imrandom = rand(size(im));
%Hacemos la comparativa de la imagen igual que en el apartado 1
im(im>=imrandom) = 1;
im(im<imrandom) = 0;
figure
imshow(im2);
title("Imagen.png")
figure
imshow(im);
title("Imagen.rng")
std2(im2-im)
```




```
>> lab_1_2

ans                                                    =
```

0.4055

Sin embargo, este numero varía en cada iteración al tratarse de una imagen en el que el dithering se ha generado con números aleatorios.

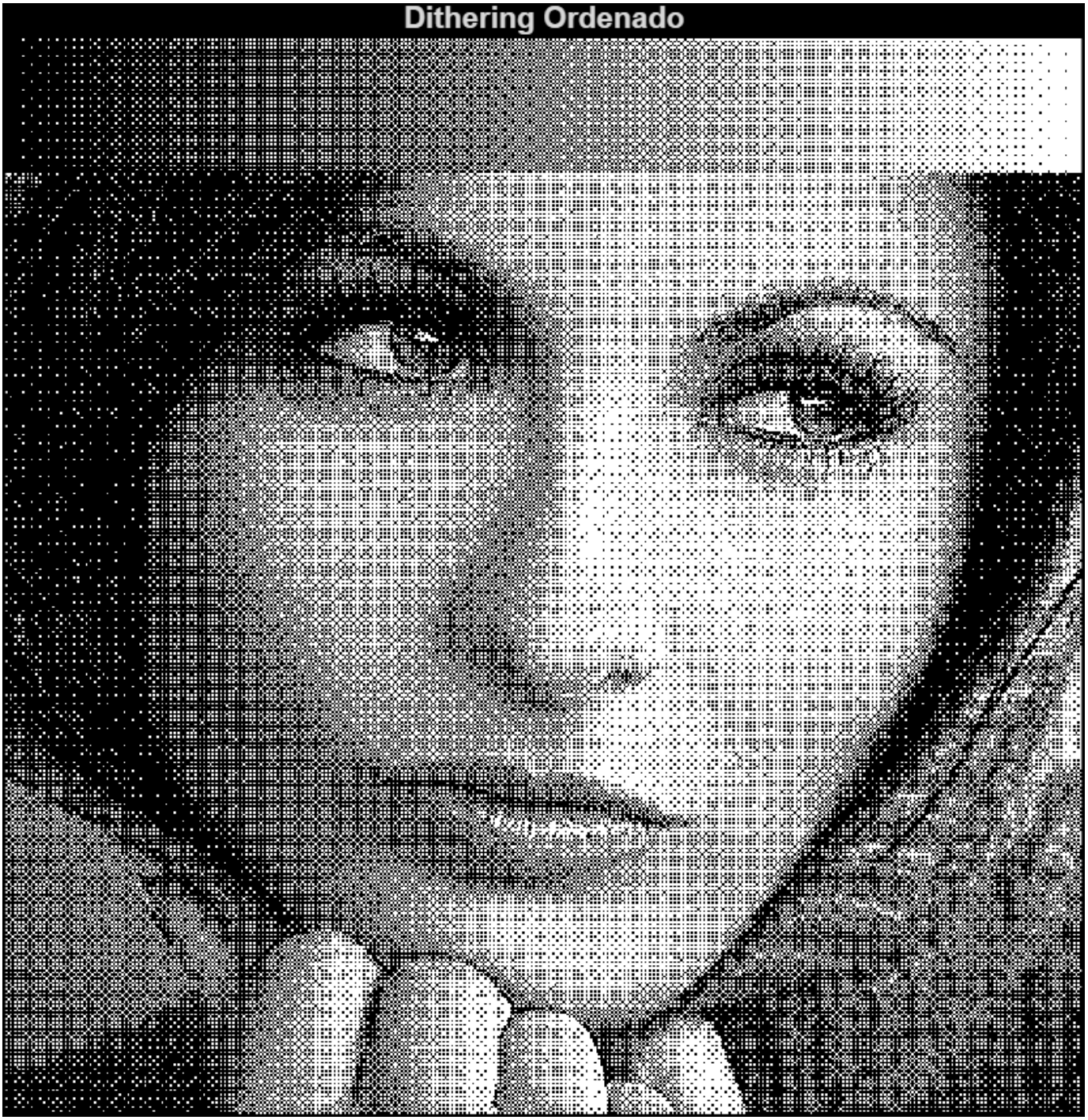
Volcad los valores de la matriz R_3 (8x8) obtenida con este algoritmo.

```
ans                                                                =

    0.0078    0.5078    0.1328    0.6328    0.0391    0.5391    0.1641    0.6641
    0.7578    0.2578    0.8828    0.3828    0.7891    0.2891    0.9141    0.4141
    0.1953    0.6953    0.0703    0.5703    0.2266    0.7266    0.1016    0.6016
    0.9453    0.4453    0.8203    0.3203    0.9766    0.4766    0.8516    0.3516
    0.0547    0.5547    0.1797    0.6797    0.0234    0.5234    0.1484    0.6484
    0.8047    0.3047    0.9297    0.4297    0.7734    0.2734    0.8984    0.3984
    0.2422    0.7422    0.1172    0.6172    0.2109    0.7109    0.0859    0.5859
    0.9922    0.4922    0.8672    0.3672    0.9609    0.4609    0.8359    0.3359
```

Adjuntad el código para crear R a partir de R_3 (8x8), la imagen resultante final y el detalle alrededor del ojo derecho. Volcad también el valor de la desviación standard de la diferencia entre el original y el resultado filtrado.

```
%Algoritmo de dithering ordenado

function R = dord(n)
   M = 0;
   Y = 1;
   for i = 1:n
      Y=Y/4;
      M = [M, M+2*Y,
         M+3*Y, M+Y];
   end
   R = (M + Y/2);
end
im3 = im2;
B = repmat(dord(3),size(im3)/8);
im3(im3>=B) = 1;
im3(im3<B) = 0;
figure
imshow(im3);
title("Dithering Ordenado");
figure
zona_ojo = im3(110:220, 320:440);
imshow(zona_ojo);
title('Zona del ojo');

std2(im2-im3);
```

```
ans                                                    =

    0.4063
```

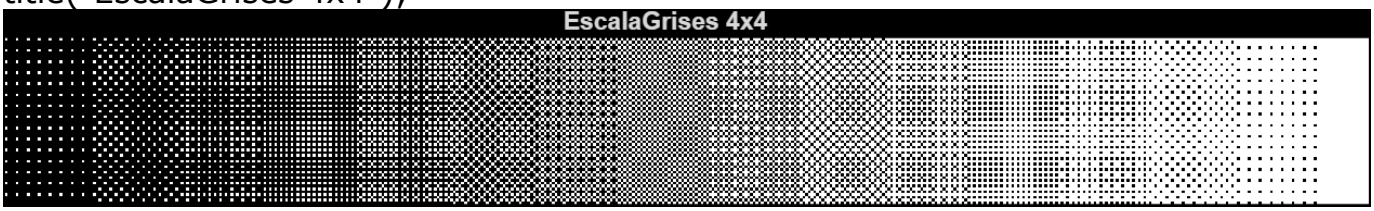


Dithering usando ruido azul:

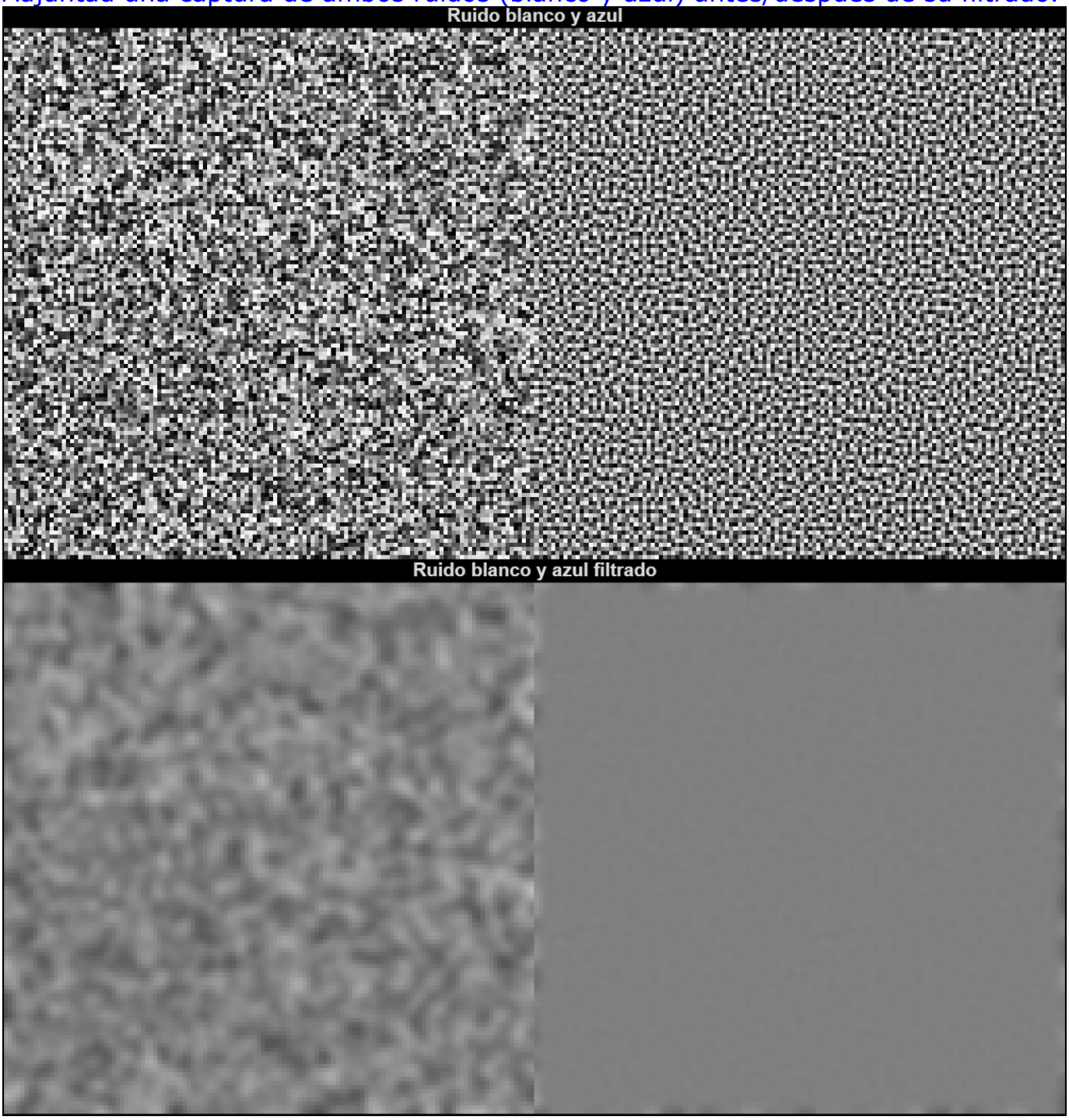
Repetir el proceso de dithering usando la matriz R_2 (4x4) y adjuntad una captura de la zona de la escala de grises en la imagen resultante.

```
im4 = im2;
M = repmat(dord(2),size(im4)/4);
im4(im4>=M) = 1;
im4(im4<M) = 0;
figure
zonagrises = im4(1:60, :);
imshow(zonagrises);
```

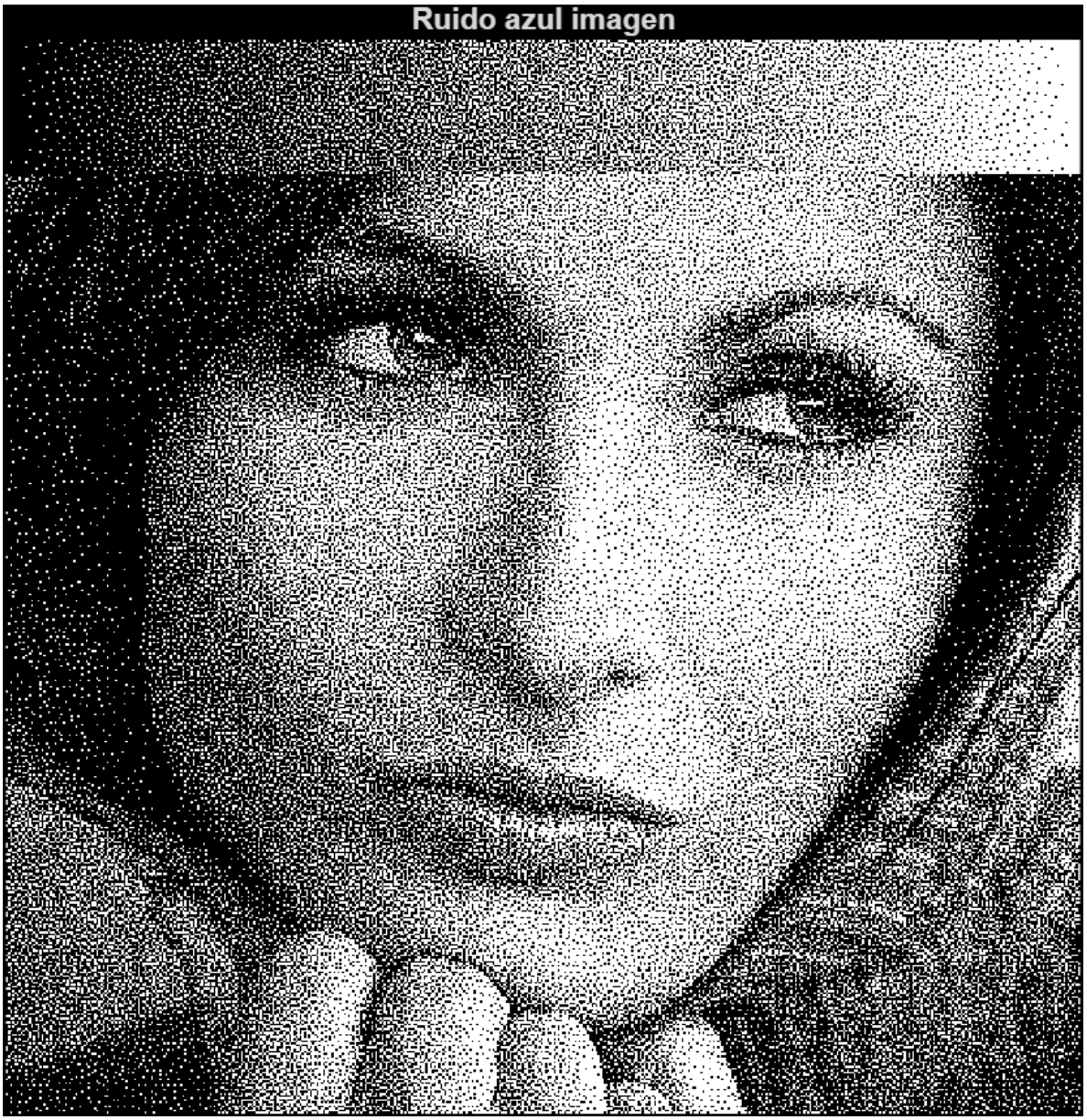
title("EscalaGrises 4x4");



Adjuntad una captura de ambos ruidos (blanco y azul) antes/después de su filtrado.



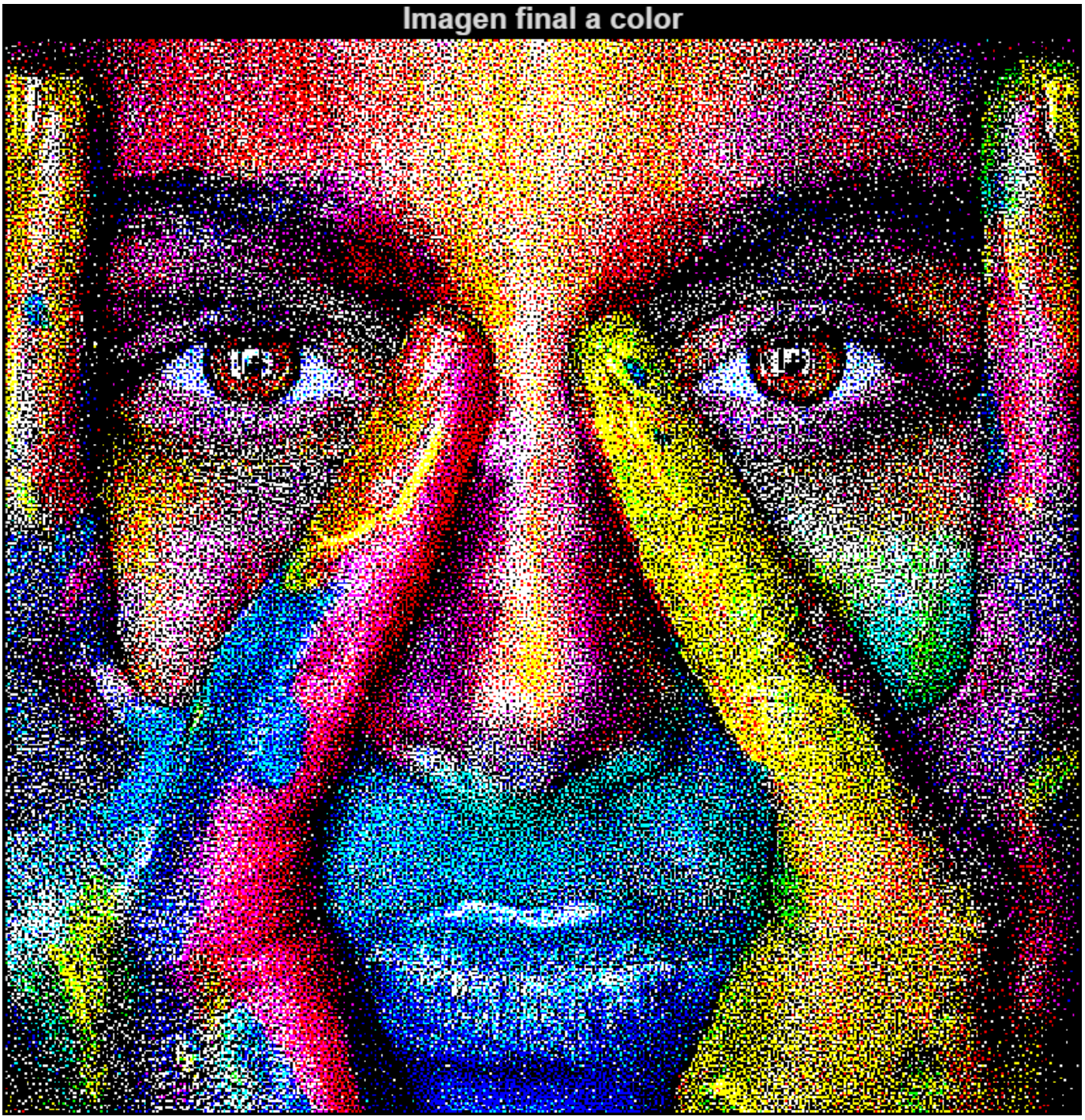
Adjuntad la imagen resultado.



Adjuntad código usado y la imagen resultado.

```
%Ruido azul imagen a color
imc = imread("FC_PROY1/color.jpg");
size(imc);
BC = repmat(B,512/64,512/64);
imc = im2double(imc);
R = imc(:,:,1);
G = imc(:,:,2);
V = imc(:,:,3);


%Aqui nos dimos cuenta de que la comparativa no tenia mucho sentido y la
%mejoramos
R = R >= BC;
G = G >= BC;
V = V >= BC;
imcfinal = double(cat(3, R, G, V));
```

```
figure
imshow(imcfinal);
title("Imagen final a color")
```



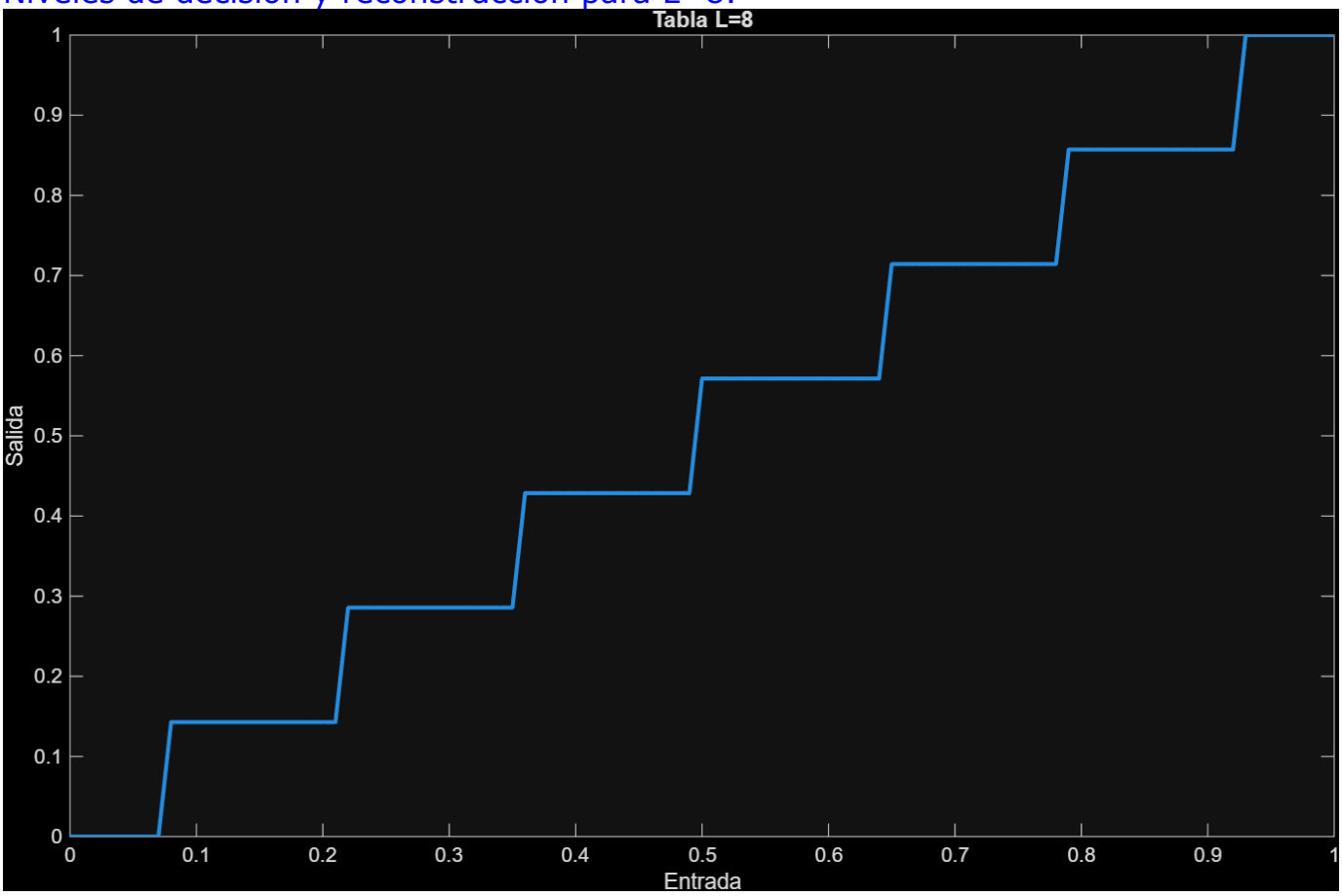
3. Dithering con más de 2 niveles

Adjuntad el código de **`function v=quant(v,L)`**

```
function x = quant(v,L)
   x = round(v*(L-1)) /(L-1);
end
```

```
L = 8;

x = [0:0.01:1]
y = quant(x,L);

figure
plot(x,y,'LineWidth',2)
xlabel('Entrada')
ylabel('Salida')
title('Tabla L=8')
```

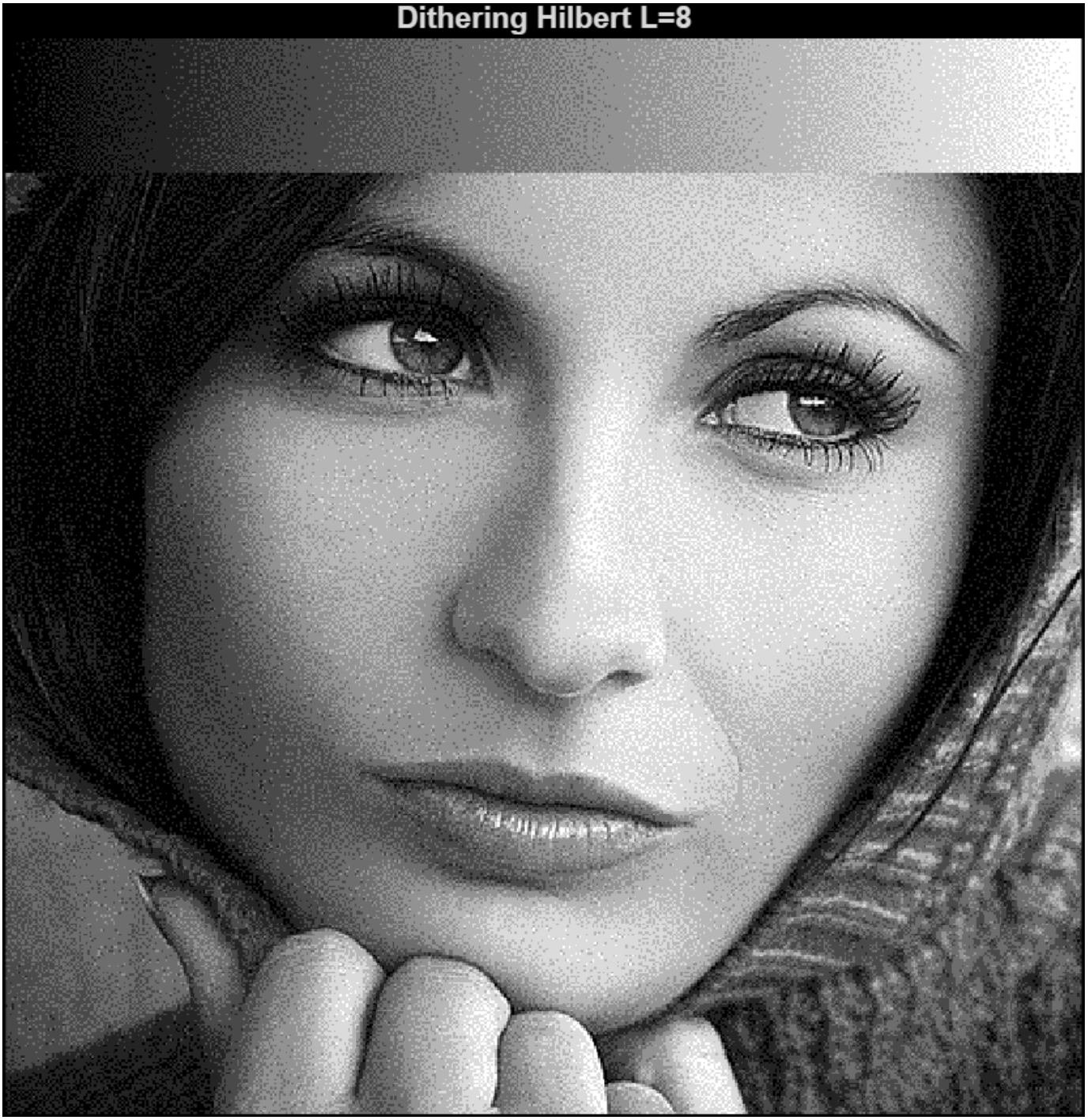
Adjuntad la gráfica obtenida para el caso de 8 niveles en la salida.
Niveles de decisión y reconstrucción para L=8.



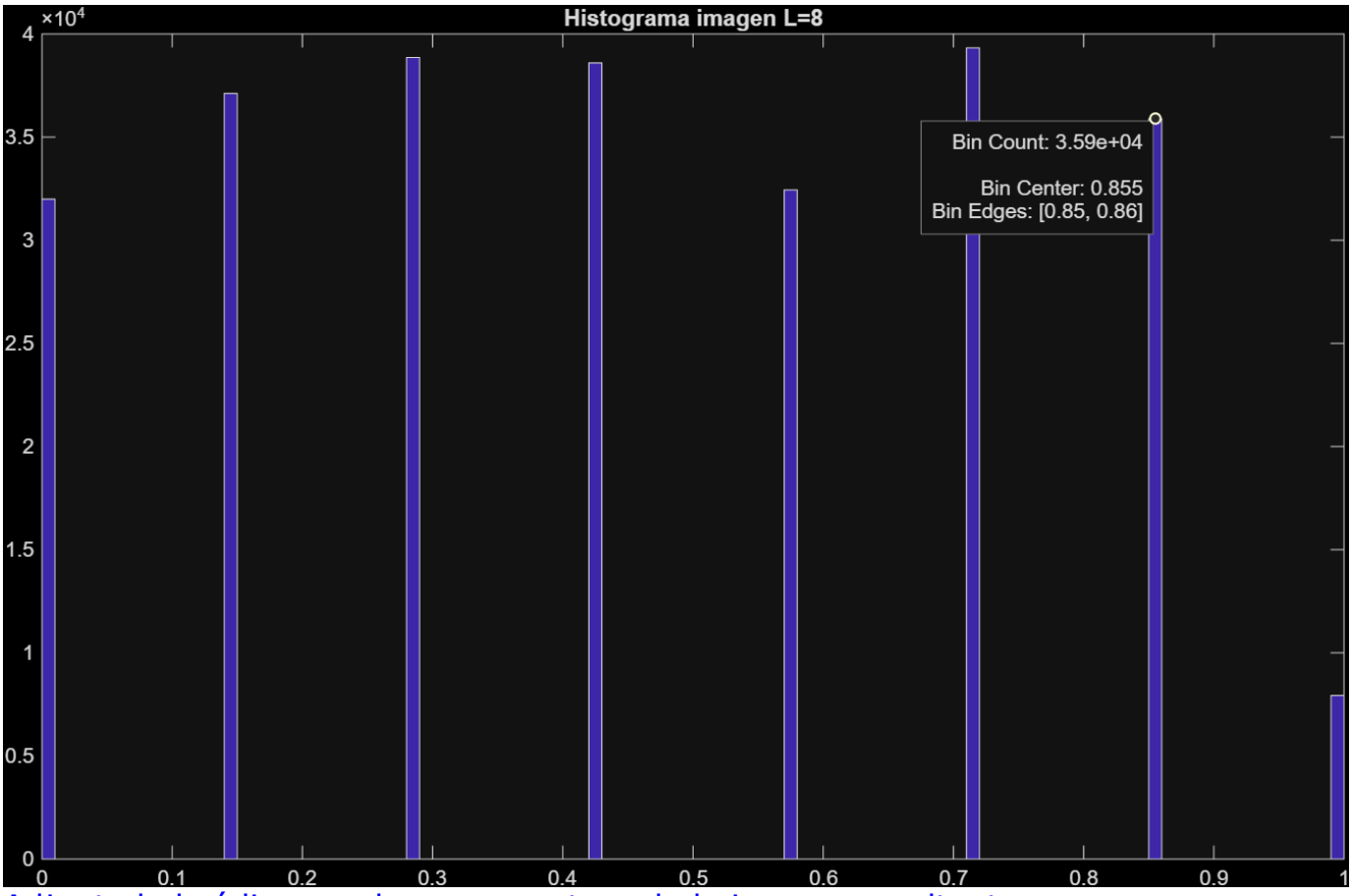
Adjuntad código de **`function im=dither_hilbert_L(im,L)`** y el resultado de aplicarla a la imagen test para el caso de L=8 niveles.

```
%Dither Hilbert para L

function im = dither_hilbert_L(im,L)

    load ("FC_PROY1/hilbert.mat"); %I,J

    im = im2double(im);
    e = 0;

    for k = 1:length(I)
```

```
        i = I(k);
        j = J(k);

        V = im(i,j) + e;

        out = quant(V,L);

        e = V - out;

        im(i,j) = out;
    end

end

im = imread('FC_PROY1/imagen.png');
im = im2double(im);

im8 = dither_hilbert_L(im,8);

figure
imshow(im8)
title('Dithering Hilbert L=8')
```

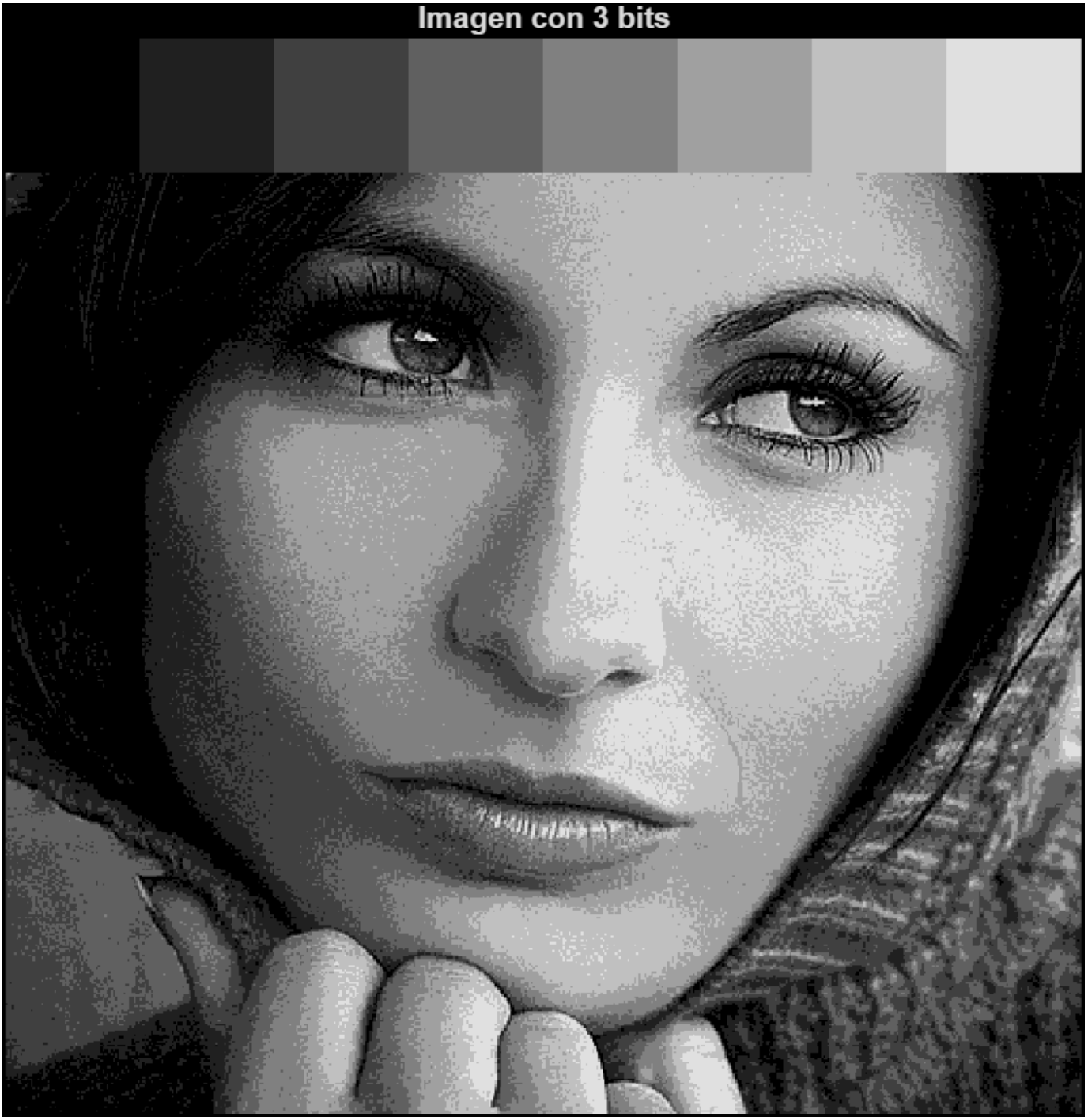


Adjuntad el histograma obtenido.



Adjuntad el código usado y una captura de la imagen resultante.

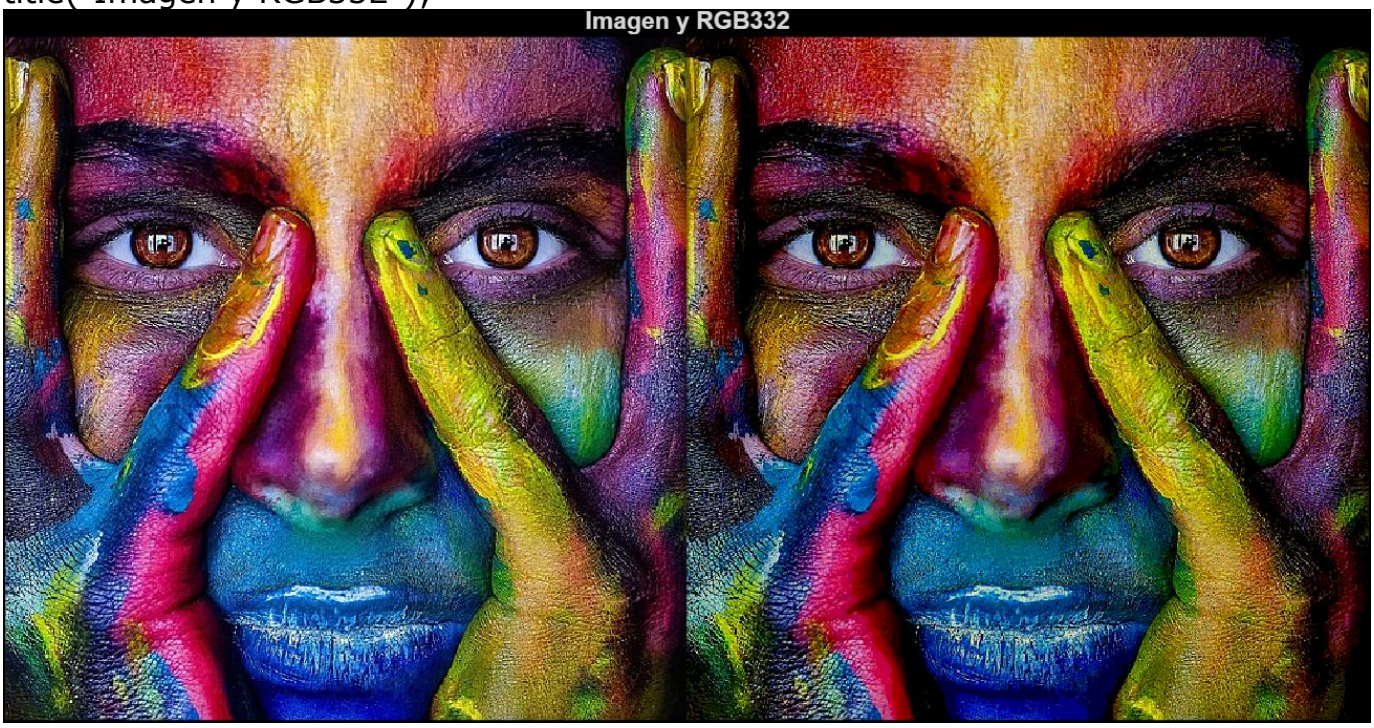
```
im_unit8 = imread('FC_PROY1/imagen.png');
im3 = bitand(im_unit8, 224);

figure
imshow(im3)
title('Imagen con 3 bits')
```



Adjuntad el código usado para obtenerla y una captura de la imagen (3+3+2) obtenida junta a la original.

```
%RGB332
imc = imread('FC_PROY1/color.jpg');

R = imc(:,:,1);
G = imc(:,:,2);
B = imc(:,:,3);

R = bitand(R, 224);
G = bitand(G, 224);
```

```
B = bitand(B, 192);

imc2 = cat(3,R,G,B);
figure
imshow([imc imc2])
title("Imagen y RGB332");
```



4. Recuperación de una imagen oculta

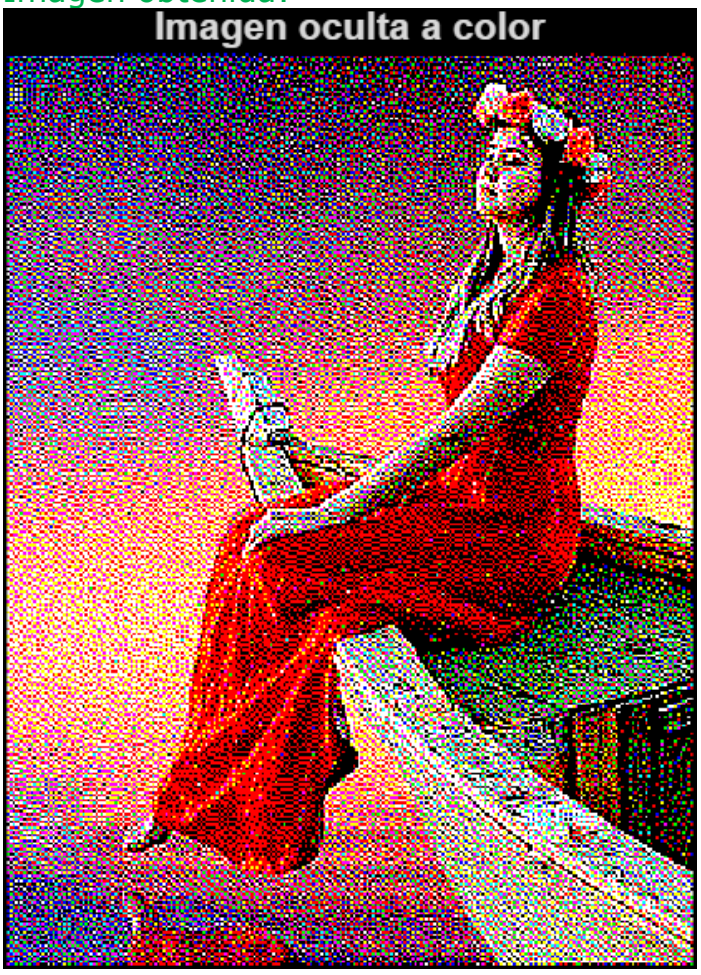
Adjuntad el código usado para que yo vea cómo lo habéis hecho y adjuntad una captura de la imagen obtenida.

Código usado:

```
% Medidas de la imagen oculta
filas = 340;
columnas = 256;
separacion = 512;
nbits = filas*columnas; % nbits por canal (según el enunciado es a coloir asiq 3 canales)
% extramos los bits de cada canal
    % - Canal Rojo -
    bits_rojo_p1 = im_oculta(1:170, 1:256);
    bits_rojo_p2 = im_oculta(1:170, 257:512);
    Rojo = [bits_rojo_p1; bits_rojo_p2];
    % - Canal Verde -
    bits_verde_p1 = im_oculta(172:341, 1:256);
    bits_verde_p2 = im_oculta(172:341, 257:512);
    Verde = [bits_verde_p1; bits_verde_p2];
    % - Canal Azul -
    bits_azul_p1 = im_oculta(343:512, 1:256);
    bits_azul_p2 = im_oculta(343:512, 257:512);
```

```
    Azul = [bits_azul_p1; bits_azul_p2];
% combinamos los canales para obtener la imagen oculta a color
imagen_oculta_color = cat(3, Rojo, Verde, Azul);
% Mostramos la imagen oculta a color
figure;
imshow(imagen_oculta_color*255);
title('Imagen oculta a color');
```

Imagen obtenida:



Destacar que para realizar esto nos hemos guiado en parte usando el mostrado de imagen oculta del lab 0 con el que hemos podido distinguir visualmente la distribución exacta que tenían los distintos canales de la imagen oculta

Código usado para esto haciendo referencia al lab0:

```
% ===== Mostrar imagen oculta al igual que con cervino.png del lab0 =========
clear; clc; close all;

im = imread('FC_PROY1/imagen.png');
% Extraemos el bit menos significativo
V =1;
im_oculta = bitand(im, V);
% Mostramos la imagen oculta igual que con el lab 0
```

```
figure;
imshow(im_oculta*255);
title('Imagen oculta');
% con esto podemos observar la imagen oculta en escala de grises que se repite 3 veces (1 por canal de color)
```

Imagen obtenida: