Laboratorio sobre sensores y revelado

Ejercicio 0:

Multiplicar por 2 el vector n1 (lo que equivale a amplificarlo). ¿Qué le pasa a su σ?
Si ampliamos el espectro de valores que pueden tomar las muestras estamos dispersando los posibles valores y por tanto estamos incrementando su desviación típica.

¿Cuál es la σ (std) de la suma de n1 y n2? ¿Y de su resta?
Tanto para la suma como para la resta la desviación típica es de 1.4

Ejercicio 1 (ruido de lectura):

b) Repetir para extraer los demás colores. Adjuntar código utilizado.

```
im = imread(black.pgm);
R = im(1:2:end,1:2:end);
G1 = im(1:2:end,2:2:end);
G2 = im(2:2:end,1:2:end);
B = im(2:2:end,2:2:end);
```

c)

	R	G1	G2	B
Media (m)	128.0076	128.0078	127.9375	127.8329
Sigma (σ)	2.6025	1.8027	1.8493	2.7528

d) Visualizar un histograma (usando fc_imhist) de uno de los canales. ¿Creéis que esta cámara está usando un offset para conservar los valores "negativos" del ruido?
Sí que está aplicando un offset porque ha convertido los datos desde la zona de 0 que debería obtener a una zona con media en 128º.
¿Cuál sería el valor del cuantificador correspondiente a un negro en esta cámara?
Un negro puro debería de tener un valor de 0 en los tres canales.

e) Observando las σ's de los 4 canales ¿notáis alguna diferencia? Dado que los "sensels" de cada canal son básicamente idénticos ¿qué puede estar haciendo la cámara para provocar esas diferencias?
Los sensores encargados del canal verde no son amplificados por el mismo factor que el resto de canales. Una hipótesis de la razón de hacer esto es que el fabricante pretenda resolver por hardware (en la etapa de amplificación) la complicación de que haya el doble de superficie de sensores verdes que del resto, o que el resto de sensores tenga filtros que no dejan pasar tanta cantidad de luz haya que compensarlo.

Ejercicio 2 (Balance de blancos):

Extraer un rectángulo de la imagen en la zona del papel, justo encima de las fichas roja/amarilla con coordenadas Y≅(650-700) y coordenadas X≅(650-750) Visualizarlo para ver que habéis escogido bien la zona. Visto aisladamente, ¿se ve blanco?

Tienen predominancia los colores cálidos.

Adjuntar vuestro código y los valores obtenidos para el vector rgb y el vector de correcciones.

```
% Limpieza
clear all;
close all;

%Carga de datos
im = imread('color.jpg');
image(im);
% fc_truesize
recorte = im(650:1:700,650:1:750,:);
% image(recorte);
R = recorte(:,:,1);
G = recorte(:,:,2);
B = recorte(:,:,3);

vectorRGB= [mean2(R);mean2(G);mean2(B)];
m = mean(vectorRGB);
c = m./vectorRGB
im2 = double(im);
im2(:,:,1) = im(:,:,1).*c(1);
im2(:,:,2) = im(:,:,2).*c(2);
im2(:,:,3) = im(:,:,3).*c(3);
im2 = uint8(im2);
figure;
image(im2);
```

Vector RGB:

0.8435
0.9853
1.2507

Adjuntar la imagen original y la resultante.

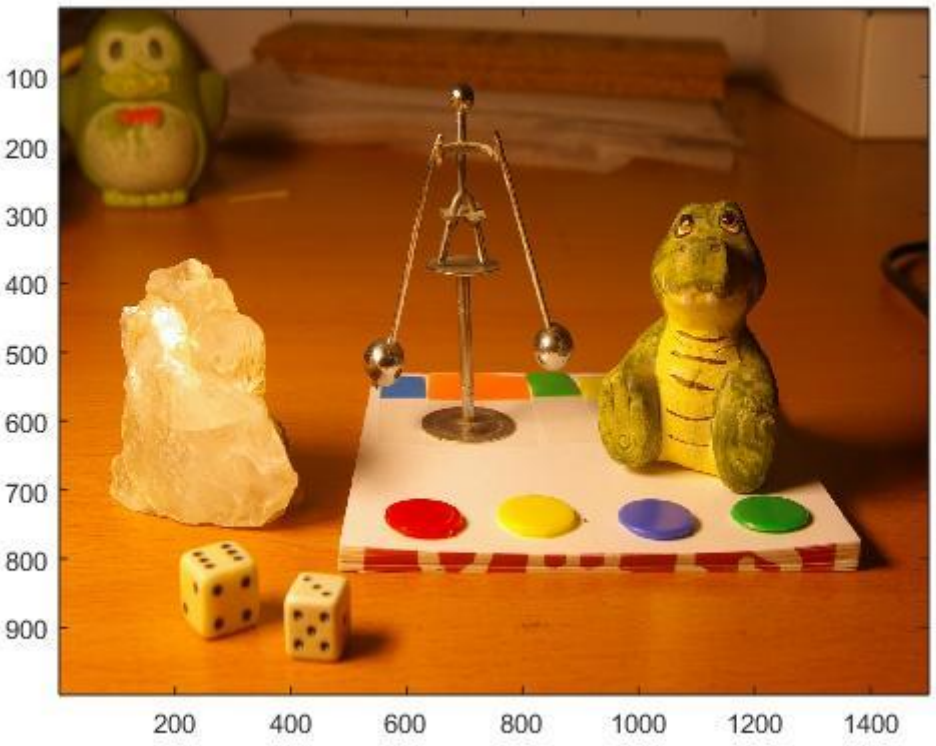


Imagen original

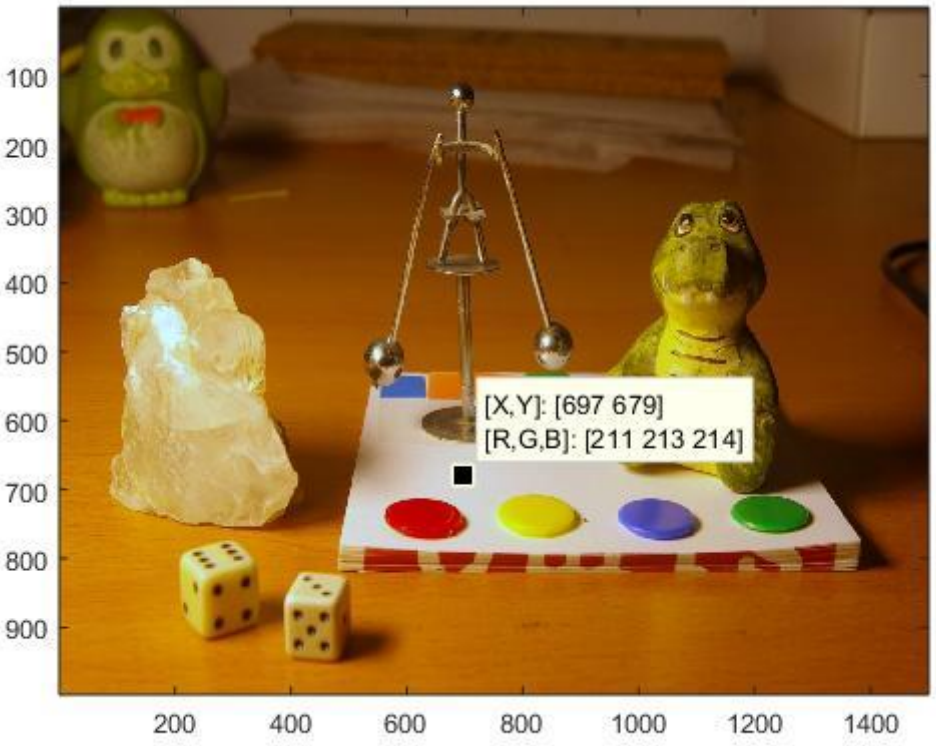


Imagen corregida

Ejercicio 3:

Adjunta código y gráfica de ajuste

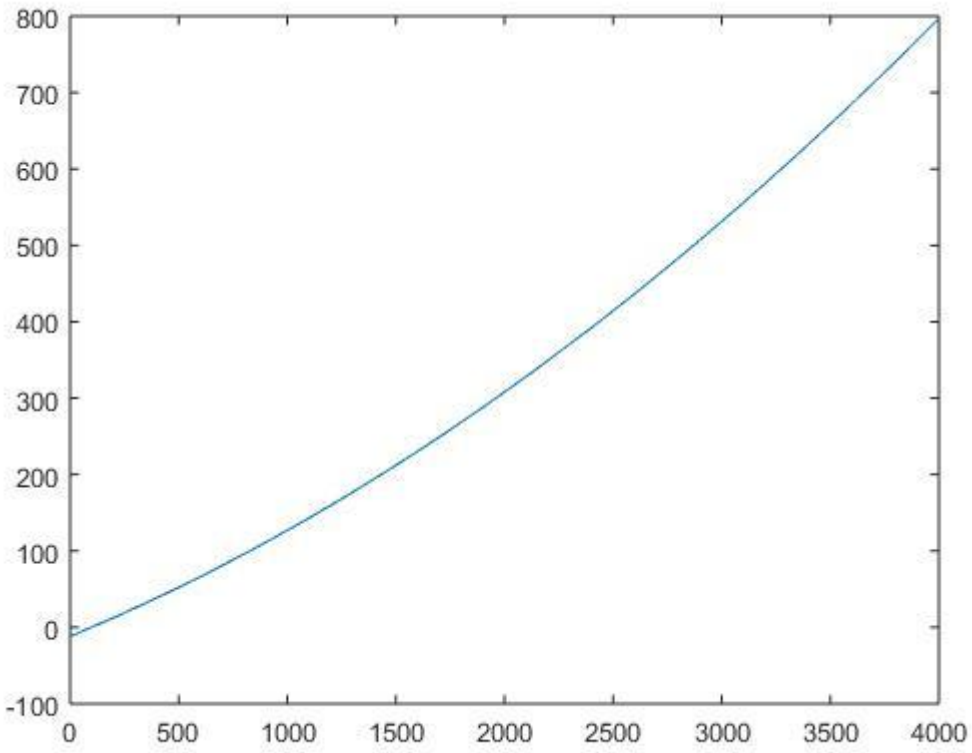
```
% Limpieza
clear all;
close all;

%Carga de datos
load datos_ruido;

E = zeros(11,1);
S = E;

for i= 1: 11
    Q=datos(:,:,i);
    E(i) = mean2(Q);
    S(i) = std2(Q)^2;
end

E2 =E.^2;
H = [E.^0 E E.^2];
c = H\S;
ee=(0:4000);
plot(ee,c(1) + c(2)*ee + c(3)*ee.^2);
```



¿Qué valor de σ se obtiene para el ruido de lectura? ¿Es similar a la que obtuvimos usando la técnica del black–frame para el canal verde?

¿Cuántos fotones necesita esta cámara para incrementar un nivel del ADC?

¿Cuántos fotones pueden caer sobre cada elemento de este sensor antes de que se llene?

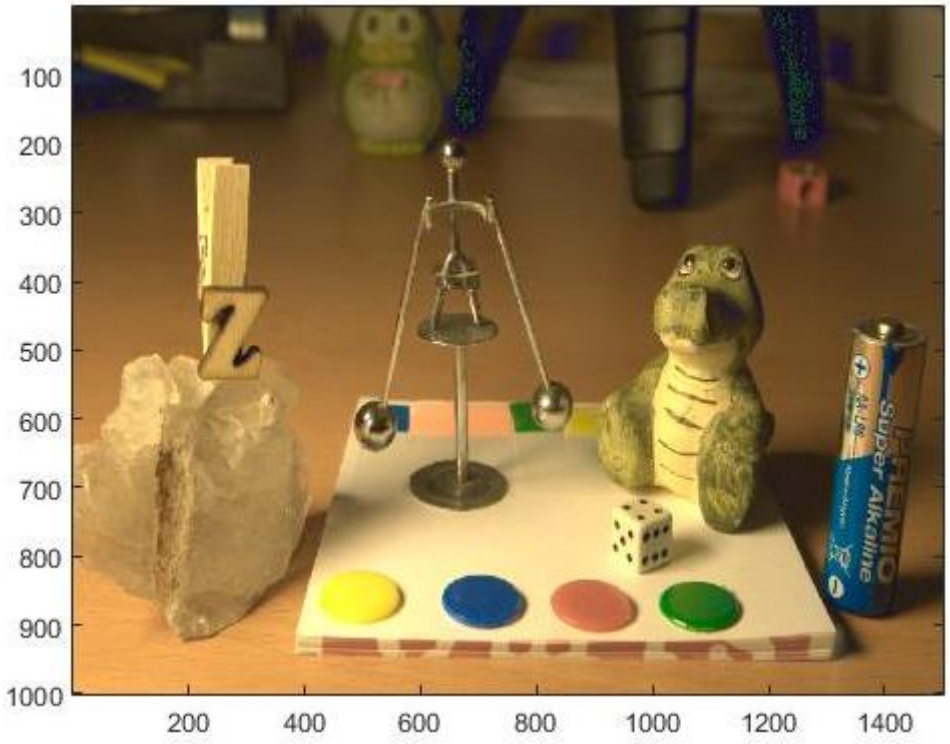
¿De qué orden (expresarla en %) es la constante K que indica la no-uniformidad de los receptores?

Consideremos una exposición del orden de 2000 (en unidades ADUs), lo que para esta cámara (con un conversor de 12 bits $\cong$ 4000 niveles) corresponde a una exposición media ¿cuáles son las σ's de los tres tipos de ruidos considerados? ¿Cuál es el más importante?

Proyecto 3

Paso 5. Visualizar la imagen con imshow. Adjuntar imagen:

Paso 8. Visualizar la imagen final con image(). Adjuntar la imagen resultante.



Paso 9. ¿Qué tamaño tiene la imagen guardada en disco?
La imagen acaba teniendo un tamaño de 216 KB.

ADJUNTAR CÓDIGO de TODO EL PROCESO.

```
% Limpieza
clear all;
close all;

%Carga de datos
im = imread('raw_red.pgm');
im = double(im);
c=[0.75 0.75 1.0]; % Balance de blancos antiguo


% Adecuación de datos de imagen
im(im < 128) = 128;
im = im-128;
im = im./(4096-128);

% Creamos matrices para los canales
R = zeros(size(im));
G = R;
B = R;
R(1:2:end,1:2:end) = im(1:2:end,1:2:end);
G(1:2:end,2:2:end) = im(1:2:end,2:2:end);
G(2:2:end,1:2:end) = im(2:2:end,1:2:end);
B(2:2:end,2:2:end) = im(2:2:end,2:2:end);

%
% 751 814
% 633 787
% Extra Balance de blancos
mediaR = mean2(R(751:2:814,631:2:780));
mediaB = mean2(B(750:2:814,630:2:780));
mediaG = mean([mean2(G(750:2:814,631:2:780));mean2(G(751:2:814,630:2:780))]);
vectorRGB= [mediaR;mediaG;mediaB]
m = mean(vectorRGB)
c = m./vectorRGB
c = c'
% Aplicamos balance de blancos
R = R.*c(1);
G = G.*c(2);
B = B.*c(3);

% % Qué tenemos ahora?
% imagen1 = zeros(size(im,1),size(im,2),3);
% imagen1(:,:,1) = R;
% imagen1(:,:,2) = G;
% imagen1(:,:,3) = B;
% imshow(imagen1);

% Interpolacion
rojo = [1/4,1/2,1/4;1/2,1,1/2;1/4,1/2,1/4];
verde = [0,1/4,0;1/4,1,1/4;0,1/4,0];
Ri = conv2(R,rojo,'same');
Gi = conv2(G,verde,'same');
Bi = conv2(B,rojo,'same');

% Creacion de imagen
imagen2 = zeros(size(im,1),size(im,2),3);
```

```
imagen2(:,:,1) = Ri;
imagen2(:,:,2) = Gi;
imagen2(:,:,3) = Bi;
imagen2 = imagen2(2:end-1,2:end-1,:); % Eliminamos el borde de píxeles con los que no
se puede hacer media
figure;
imshow(imagen2);

% Paso a sRGB
M = [0.6844 0.1651 0.1009;0.3600 0.7634 -0.1235;0.0389 -0.0575 1.1072]*[3.2406 -
1.5372 -0.4986;-0.9689 1.8758 0.0415;0.0557 -0.2040 1.0570];
for y = 1:size(imagen2,1) %Esto es terriblemente lento y tiene que haber una mejor
forma de hacerlo
    for x = 1:size(imagen2,2)
        vector = [imagen2(y,x,1);imagen2(y,x,2);imagen2(y,x,3)];
        vector = M * vector;
        imagen2(y,x,1) = vector(1);
        imagen2(y,x,2) = vector(2);
        imagen2(y,x,3) = vector(3);
    end
end
imagen2(imagen2 < 0)= 0;
imagen2(imagen2 > 1) = 1;
% figure;
% imshow(imagen2);

% Aplicamos Gamma
imagen2(imagen2<0.0031308) = imagen2(imagen2<0.0031308)*12.92;
imagen2(imagen2>=0.0031308) = (imagen2(imagen2>=0.0031308).^(1/2.4))*1.055-0.055;
% figure;
% imshow(imagen2);

% EXTRA 2 Incrementamos saturación
imagenSAT08 = saturacion(imagen2,0.5);

% Volvemos a 8bits
imagen2 = imagen2*255;
imagen2 = uint8(imagen2);

imagenSAT08 = imagenSAT08*255;
imagenSAT08 = uint8(imagenSAT08);

% Retocamos un poco
imagenSAT08 = imagenSAT08*1.2;
imagen2 = imagen2 *1.2; % Brillo
figure;
image(imagen2);

% Guardamos como jpg
imwrite(imagen2,'resultado.jpg','Quality',90);
imwrite(imagenSAT08,'imagenSAT08.jpg','Quality',90);
```

FUNCIÓN SATURACION

```
function [ imagenOut ] = saturacion( imagenIn,p )

imagenOut = rgb2hsv(imagenIn);
imagenOut(:,:,2) = imagenOut(:,:,2).^p;
imagenOut = hsv2rgb(imagenOut);

end
```

EXTRAS:

Balance de blancos:

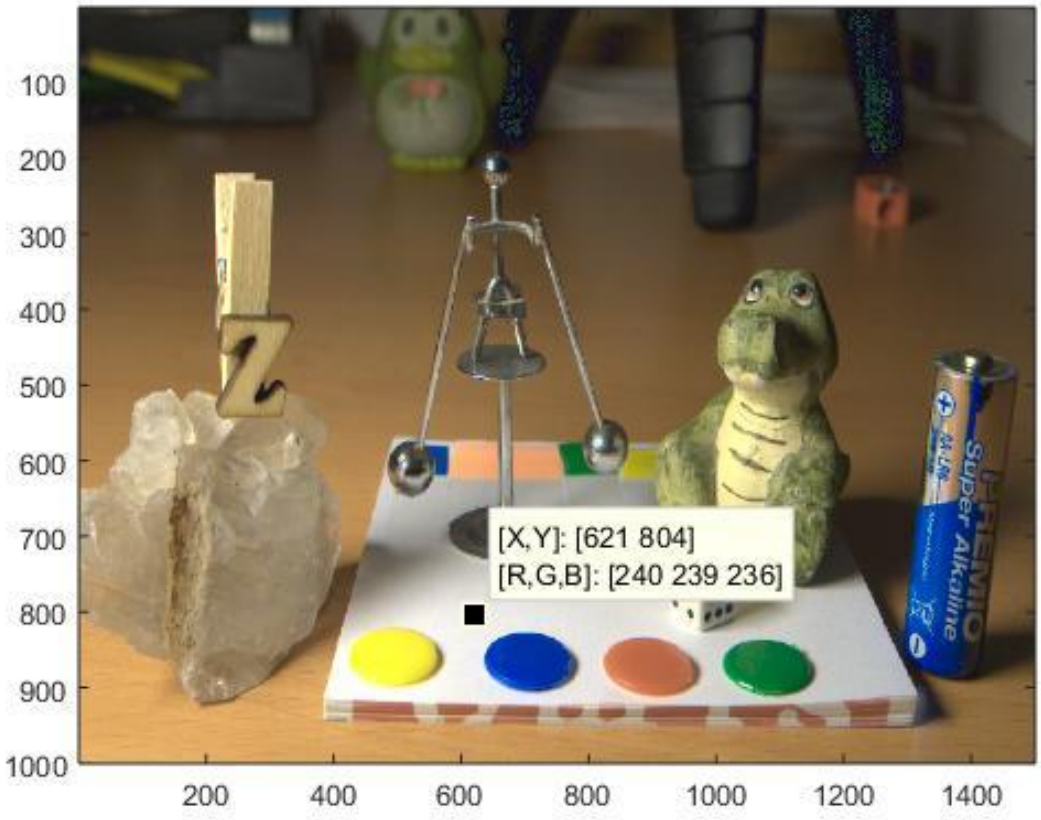


Imagen final tras corregir balance de blancos

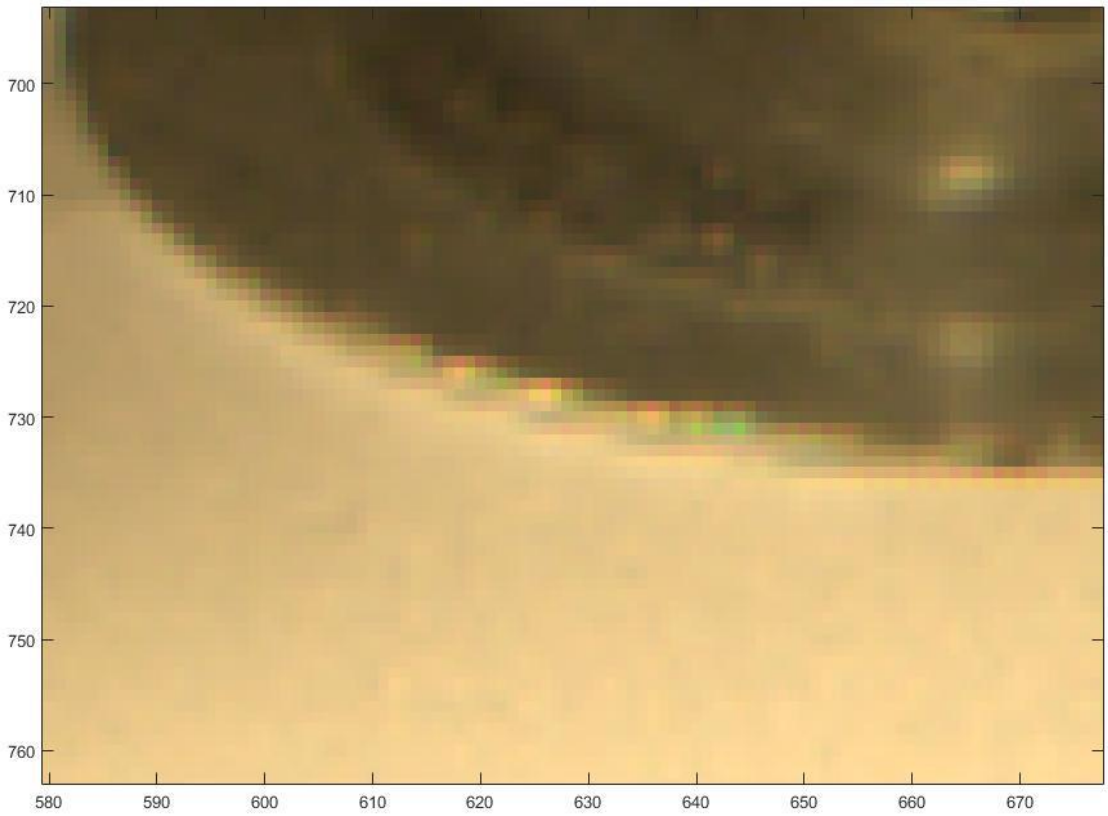
Saturación:

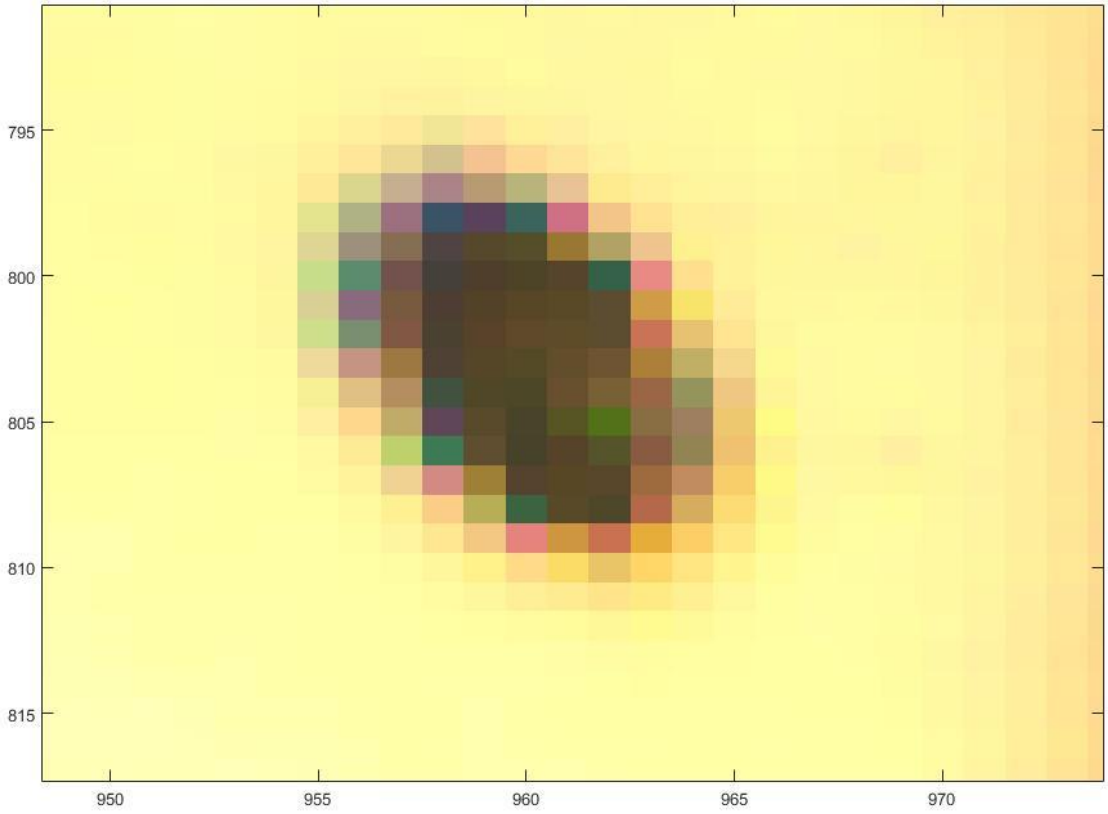
Imagen final con saturación 0.8

¿En qué zonas en la imagen pueden aparecer efectos debidos al algoritmo de "demosaicing" ingenuo que hemos usado? Detectar alguna de esas zonas y hacer zoom sobre ellas.

Los efectos de aplicar un demosaicing malo se encuentran en las zonas de alta frecuencia de la imagen, donde los sensores contiguos cambian de valor muy fuertemente y el algoritmo da estos valores tan dispares a los pixeles de alrededor.

En las dos imágenes de abajo podemos ver que es en los bordes de los objetos donde están las altas frecuencias de la imagen, y por ello se perciben los efectos de los que hablamos.


700
710
720
730
740
750
760
580
590
600
610
620
630
640
650
660
670


795
800
805
810
815
950
955
960
965
970