FACULTAD DE INGENIERÍA
UNIVERSIDAD DE BUENOS AIRES

67.61 - FUNDAMENTOS MATEMÁTICOS DE LA VISIÓN ROBÓTICA

Esa cosa llamada sensor

CAMILA SERRA (97422)

MARIANO EZEQUIEL WILLINER (83469)

8 de junio de 2021

Introducción

El objetivo de este trabajo es tomar una foto con el celular tapando el sensor (se obtiene una imágen negra). Luego, hacer un análisis estadístico del ruído. También, a partir una foto con el celular de un patrón de barras generado, obtener el MTF.

1. Análisis estadístico del ruido

[1]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from glob import glob
%matplotlib inline
```

[2]:
```
imgs = []
img_fnames = glob('./imgs-ruido/*.jpg')
for fname in img_fnames:
    img = cv2.imread(fname)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    if img.shape[0] > img.shape[1]:
        img = cv2.transpose(img)
    imgs.append(img)

print("Se cargaron {0} imágenes".format(len(imgs)))
```

```
Se cargaron 10 imágenes
```

[3]:
```
print("Menor intensidad =", np.min([imgs]))
print("Mayor intensidad =", np.max([imgs]))
```

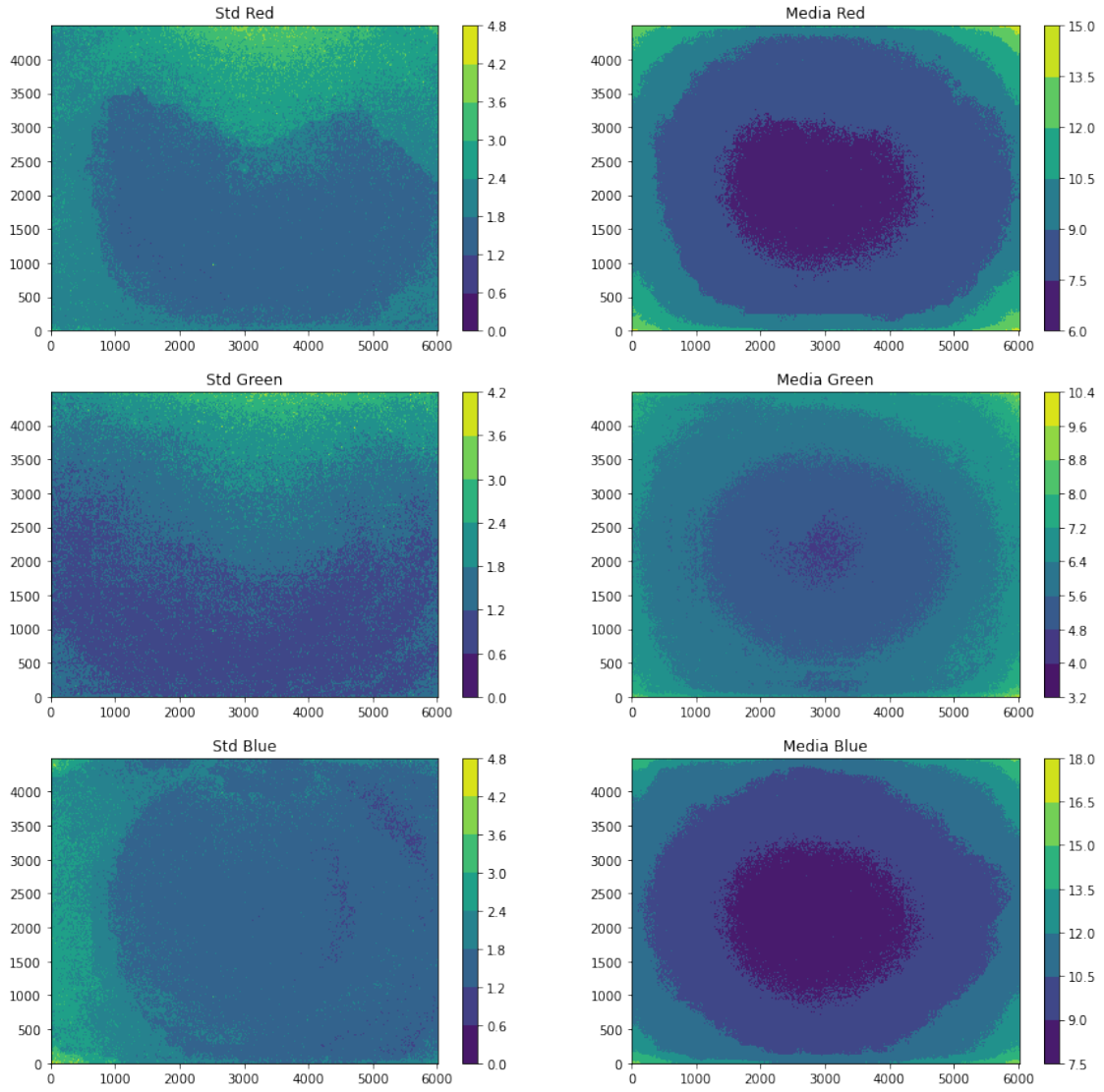
```
Menor intensidad = 1
Mayor intensidad = 30
```

[4]:
```
## media y desvio std
imgs_np = np.stack(imgs)
img_media = np.mean(imgs_np, axis=0)
img_std = np.std(imgs_np, axis=0)
```

[5]:
```
def dibujar_contorno(title, axs, mat):
    X, Y = np.meshgrid(range(len(mat[0])), range(len(mat)))
    Z = mat

    # decimación para no matar la compu calculando contornos!
    dec = 16

    cp = axs.contourf(X[::dec], Y[::dec], Z[::dec])
```

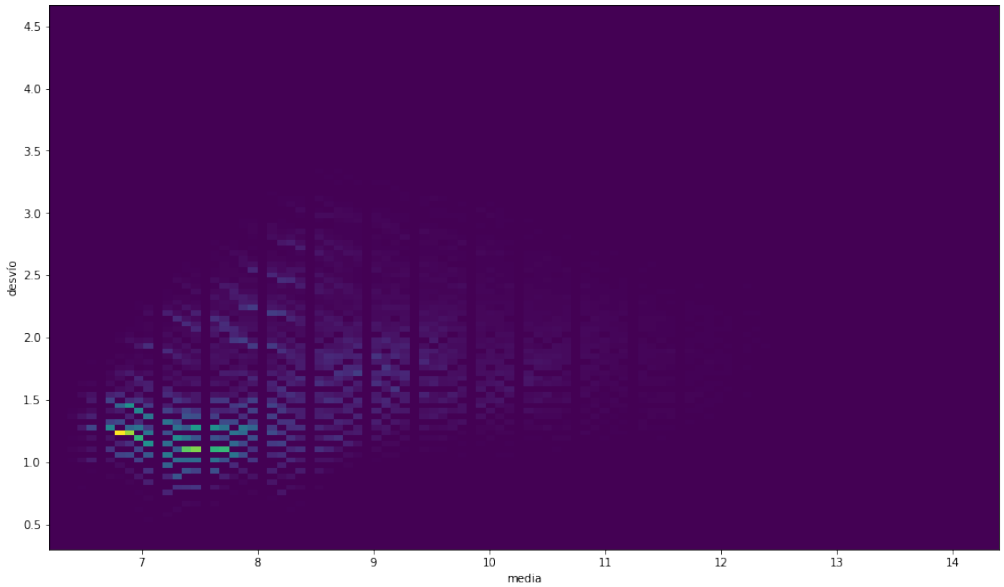
```
    axs.set_title(title)
    fig.colorbar(cp, ax=axs)
```

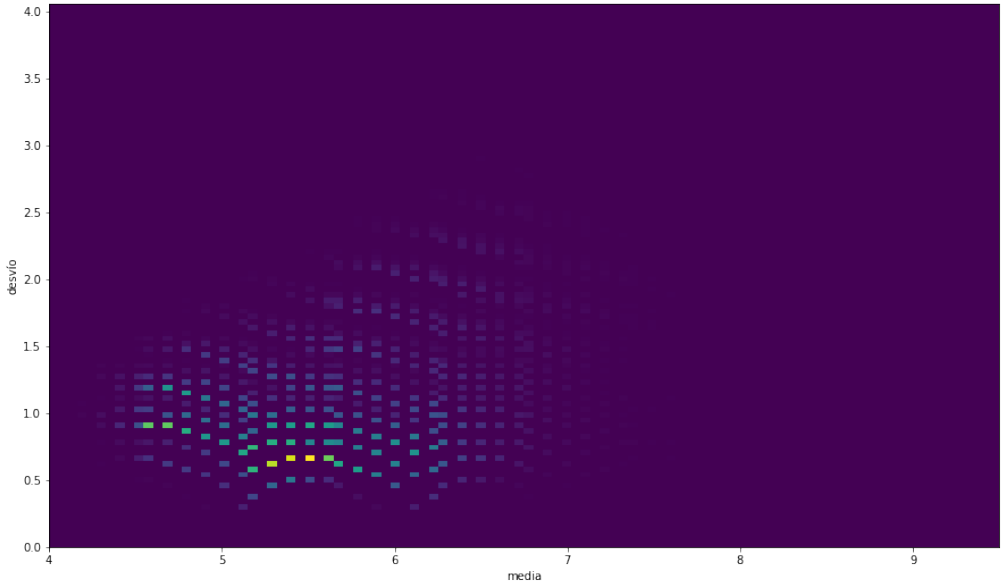
[6]:
```
fig, axs = plt.subplots(3, 2,figsize=(15,15))

for channel, number in [('Red',0), ('Green',1), ('Blue',2)]:
    dibujar_contorno('Std ' + channel,axs[number,0], img_std[:,:,number])
    dibujar_contorno('Media ' + channel, axs[number,1], img_media[:,:
 ↪,number])

plt.show()
```

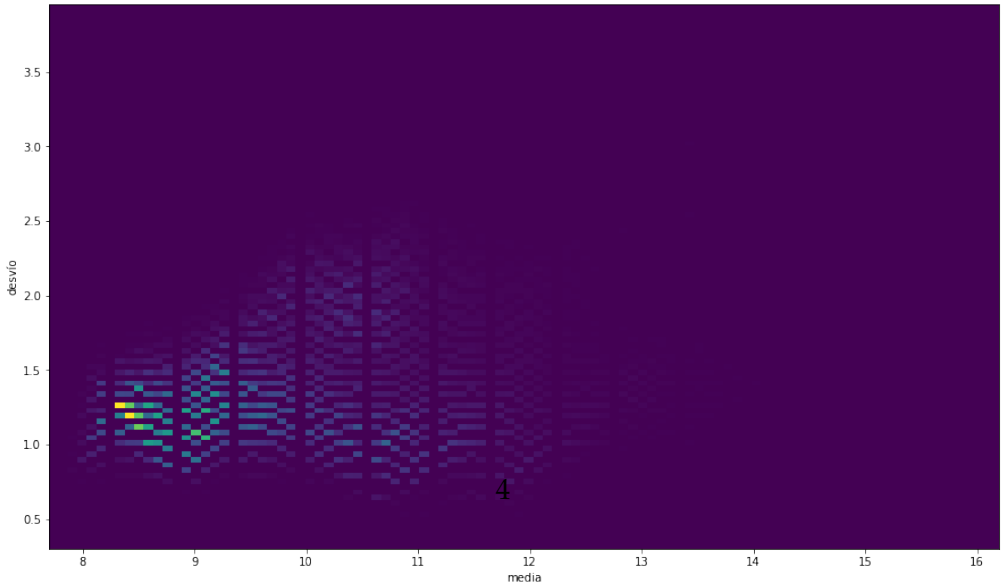


2. Estadísticas de ruido: relación entre media y desvío

[7]:
```
fig, axs = plt.subplots(3, 1, figsize=(15,30))

dec = 100

todos_los_rojos_std = np.ravel(img_std[:,:,0])
todos_los_rojos_media = np.ravel(img_media[:,:,0])

todos_los_verdes_std = np.ravel(img_std[:,:,1])
todos_los_verdes_media = np.ravel(img_media[:,:,1])

todos_los_azules_std = np.ravel(img_std[:,:,2])
todos_los_azules_media = np.ravel(img_media[:,:,2])

axs[0].set(xlabel='media' , ylabel='desvío')
cb = axs[0].hist2d(todos_los_rojos_media[::dec], todos_los_rojos_std[::
 ↪dec], bins=100)

axs[1].set(xlabel='media' , ylabel='desvío')
cb = axs[1].hist2d(todos_los_verdes_media[::dec], todos_los_verdes_std[::
 ↪dec], bins=100)

axs[2].set(xlabel='media' , ylabel='desvío')
cb = axs[2].hist2d(todos_los_azules_media[::dec], todos_los_azules_std[::
 ↪dec], bins=100)

plt.show()
```


desvío
media
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
7
8
9
10
11
12
13
14

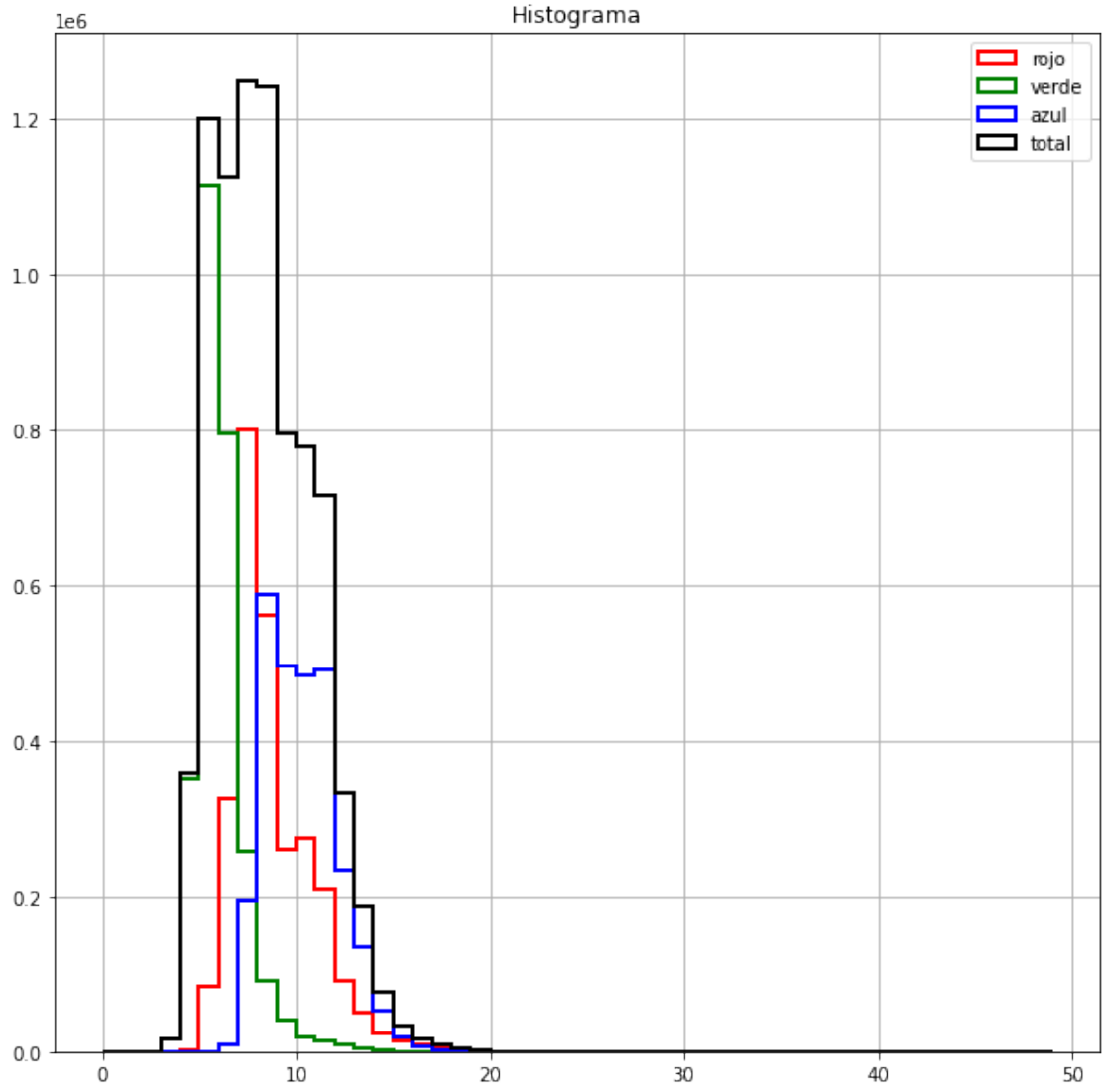

desvío
media
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
4
5
6
7
8
9


desvío
media
3.5
3.0
2.5
2.0
1.5
1.0
0.5
8
9
10
11
12
13
14
15
16

3. Histograma

[8]:
```
dec = 100
plt.figure(figsize=(10,10))
plt.title('Histograma')
todos_los_rojos = np.ravel(imgs_np[:,:,:,0])
todos_los_verdes = np.ravel(imgs_np[:,:,:,1])
todos_los_azules = np.ravel(imgs_np[:,:,:,2])
plt.grid()
i_max = 50
_ = plt.hist(todos_los_rojos[::dec], bins=range(i_max),␣
 ↪color='red',histtype='step', linewidth=2.0)
_ = plt.hist(todos_los_verdes[::dec], bins=range(i_max), color='green',␣
 ↪histtype='step', linewidth=2.0)
_ = plt.hist(todos_los_azules[::dec], bins=range(i_max), color='blue',␣
 ↪histtype='step', linewidth=2.0)

_ = plt.hist(np.ravel(imgs_np)[::dec], bins=range(i_max), color='black',␣
 ↪histtype='step', linewidth=2.0)
plt.legend(['rojo', 'verde', 'azul', 'total'])
```

[8]:
```
<matplotlib.legend.Legend at 0x7f48f02870d0>
```


Histograma
1e6
rojo
verde
azul
total
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50

[ ]:

[ ]:

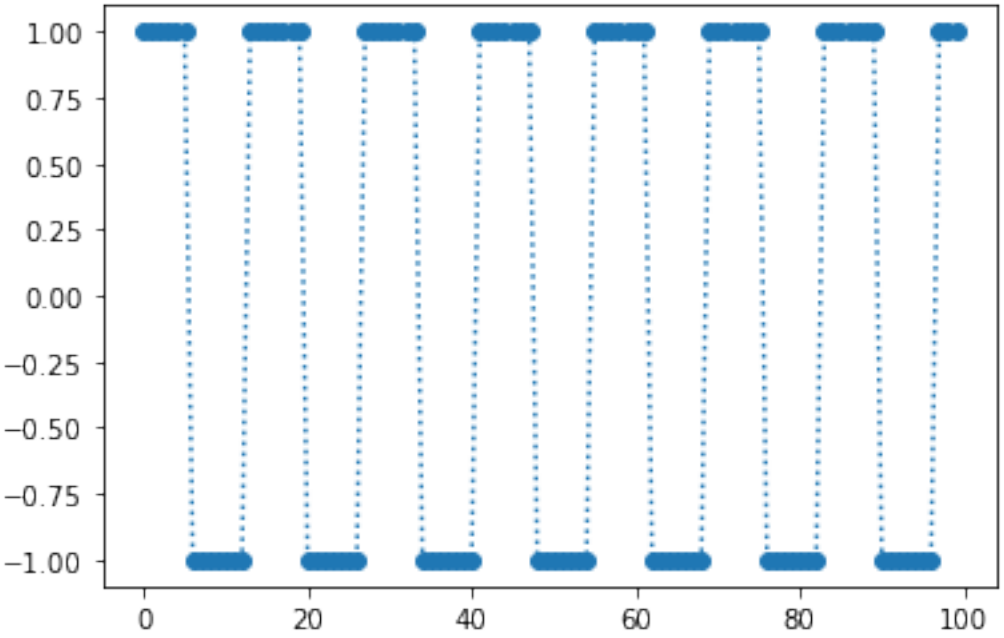
[ ]:

4. Estimación de MTF

[1]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
def generar_onda_cuadrada(largo, periodo):
    onda = np.arange(1, largo+1)
    return list(map(lambda x: -1 if (np.mod(x, periodo) >= periodo/2)␣
 ↪else 1, onda))

plt.plot(generar_onda_cuadrada(largo=100, periodo=14), ':o')
```

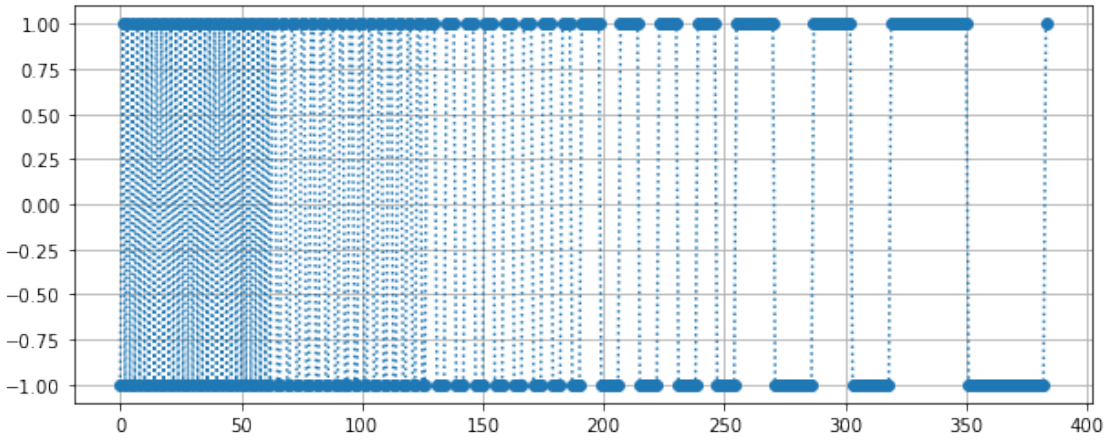
[2]:
```
[<matplotlib.lines.Line2D at 0x7fb1537fe0d0>]
```



[3]:
```
n_rep = 8
largo_max = 10
ys = []

anchos =       [ 2, 4, 8,16,32,64]
repeticiones = [32,16, 8, 4, 2, 1]

for a, r in zip(anchos, repeticiones):
    ys.append(generar_onda_cuadrada(a*r, a))
```

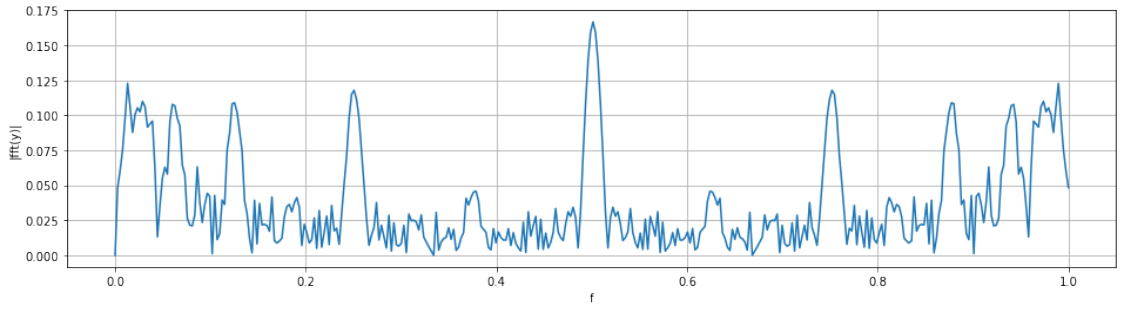
```
y = np.hstack(ys)
n_samples = len(y)

t = np.linspace(0, n_samples, n_samples)


n_show = n_samples//10
plt.figure(figsize=(10,4))
plt.grid()
plt.plot(y, ':o')
```

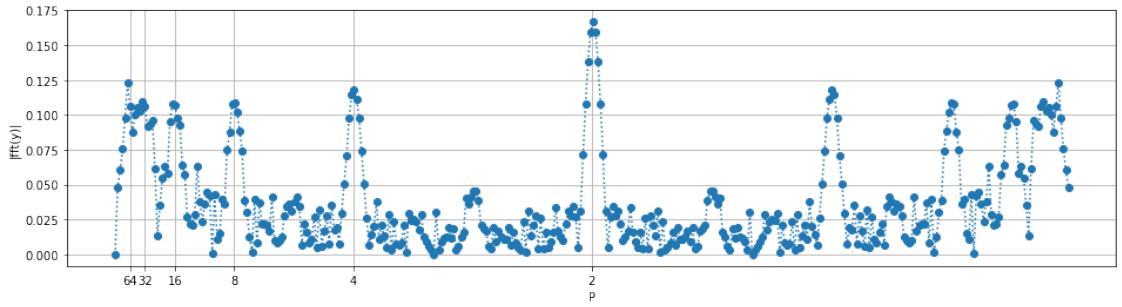
[3]: [<matplotlib.lines.Line2D at 0x7fb15376b820>]



[4]:
```
y_f = np.fft.fft(y)

ancho_pantalla = 1920
f = np.linspace(0, 1, n_samples)

plt.figure(figsize=(16,4))
plt.grid()

plt.ylabel('|fft(y)|')
plt.plot(f, np.abs(y_f)/n_samples)
plt.xlabel('f');

x = [1/(a) for a in anchos]
labels = [a for a in anchos]
plt.figure(figsize=(16,4))
plt.ylabel('|fft(y)|')
```

```
plt.grid()
n_show = n_samples
plt.plot(f[:n_show], np.abs(y_f[:n_show])/n_samples , ':o')
plt.xticks(x, labels);
plt.xlabel('P');
```



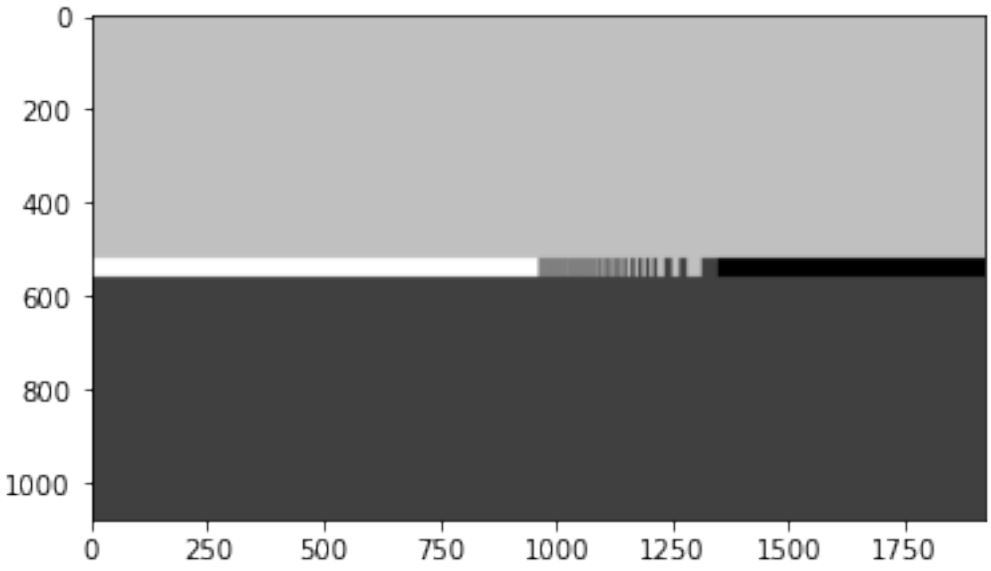


4.1. 1. Generar patrón

[5]:
```
# ajustar según pantalla donde se saca la foto, poner la resolución del monitor

ancho_pantalla = 1920
alto_pantalla = 1080

alto_franja_central_px = 40

oscuro = 64
claro = 192

medio = (claro+oscuro)/2
amplitud = (claro-oscuro)/2
```

```
onda = medio + amplitud*y

centro_izq = 255*np.ones((alto_franja_central_px, ancho_pantalla//2))
centro_der =   0*np.ones((alto_franja_central_px, ancho_pantalla//2 -␣
 ↪len(onda)))

arriba = claro*np.ones((alto_pantalla//2 - alto_franja_central_px//2,␣
 ↪ancho_pantalla))
centro = np.hstack((centro_izq, np.tile(onda, (alto_franja_central_px,␣
 ↪1)), centro_der))
abajo = oscuro*np.ones((alto_pantalla//2 - alto_franja_central_px//2,␣
 ↪ancho_pantalla))

patron_mtf = np.vstack((arriba, centro, abajo)).astype(np.uint8)

patron_mtf = cv2.cvtColor(patron_mtf, cv2.COLOR_GRAY2RGB)

plt.imshow(patron_mtf)
cv2.imwrite('patron_mtf.png', patron_mtf)
```

[5]: True

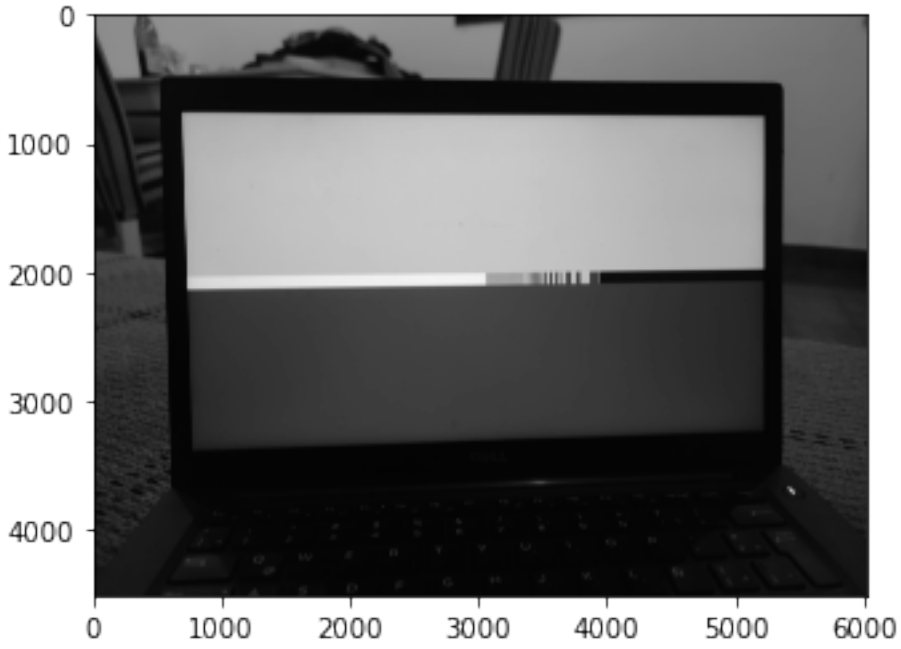


patron_mtf.png

4.2. 2. Adquirir patrón

[6]:
```
mtf_leer = cv2.imread('mtf.jpg')
mtf_leido_gray = cv2.cvtColor(mtf_leer, cv2.COLOR_BGR2GRAY)
plt.imshow(mtf_leido_gray, cmap='gray')
```

[6]:
```
<matplotlib.image.AxesImage at 0x7fb151e27be0>
```



4.3. 3. Preprocesar muestra

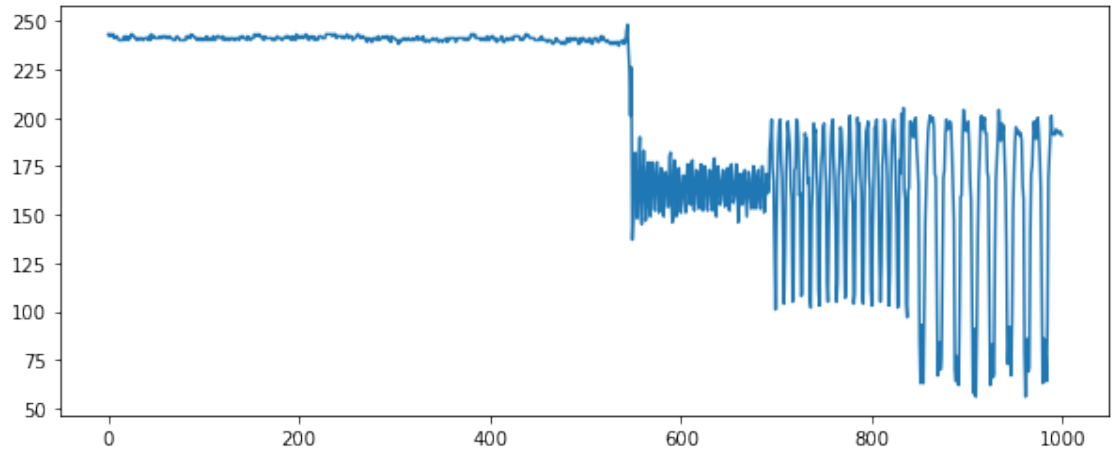
[7]:
```
## Buscar de pegarle con un corte a la zona del mtf
ajustar_linea_mtf = cv2.cvtColor(mtf_leido_gray//4, cv2.COLOR_GRAY2RGB)

ancho_foto = mtf_leido_gray.shape[1]

fila_mtf = 2070
zoom = 1600
```

```
# rellenar con principio y fin del patrón mtf en la foto:
columnas_mtf = slice(0, 5000)

ajustar_linea_mtf[fila_mtf, columnas_mtf, 0] = 255

plt.figure(figsize=(16,9));
plt.imshow(ajustar_linea_mtf);
```

[8]:
```
# refinamos el corte
zoom = 1000
fila_mtf = 2070

columnas_mtf = slice((ancho_foto//2-zoom//2), (ancho_foto//2+zoom//2))
```

[9]:
```
# dibujamos con más detalle el corte

y_est = mtf_leido_gray[fila_mtf, columnas_mtf]

plt.figure(figsize=(10,4))
plt.plot(y_est)
```

[9]: [<matplotlib.lines.Line2D at 0x7fb150e1d730>]



[10]:
```
# Generamos gráfico para leer donde cortar para recuperar la estimación␣
 ↪de la y de arriba

# por ahí si no quedó perfectamente centrado necesitan otro valor␣
 ↪diferente de +/- 35:
claro_est = mtf_leido_gray[fila_mtf-35, columnas_mtf]
oscuro_est = mtf_leido_gray[fila_mtf+35, columnas_mtf]

%matplotlib notebook
plt.figure(figsize=(10,4))
plt.plot(y_est)
plt.plot(claro_est)
plt.plot(oscuro_est)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

[10]: [<matplotlib.lines.Line2D at 0x7fb150da2190>]

[11]:
```
# Leer del gráfico de arriba
k_start = 530
k_end = 1000

y_est_r = y_est[k_start:k_end]

%matplotlib notebook
plt.figure(figsize=(10,4))
plt.grid()
```

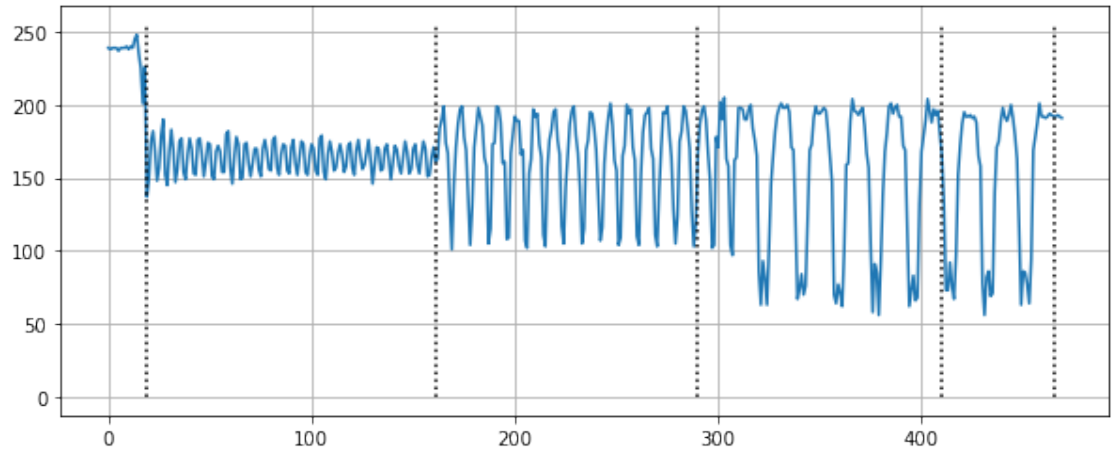
```
plt.plot(y_est_r)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

[11]:
```
[<matplotlib.lines.Line2D at 0x7fb150d5b370>]
```

[12]:
```
# llenar con bordes medidos del gráfico:
bordes= [19, 161, 290, 410, 465]

%matplotlib inline
plt.figure(figsize=(10,4))
plt.grid()
plt.plot(y_est_r)

for b in bordes:
    plt.plot([b, b], [0, 255], ':k')
```

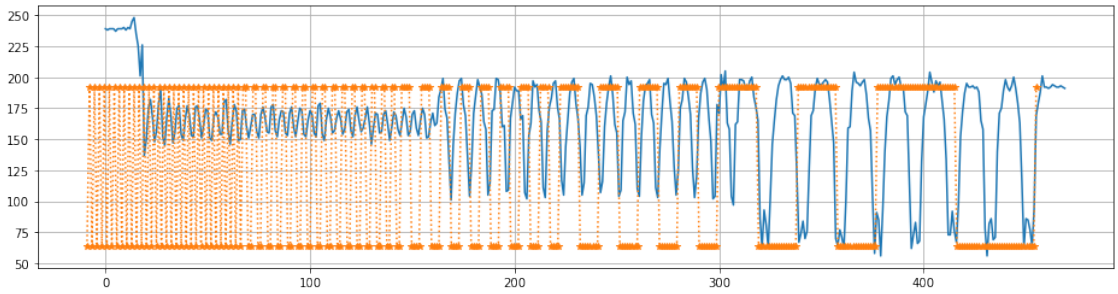


4.4. 4. Extraer MTF

[13]:
```
# comparamos con la función que le "metimos de entrada" al sistema
%matplotlib inline
plt.figure(figsize=(16,4))
plt.grid()
plt.plot(y_est_r)

# -9,455 son offsets para que quede bien, cambiarlo según lo que␣
 ↪obtuvieron
offset_l = -9
```

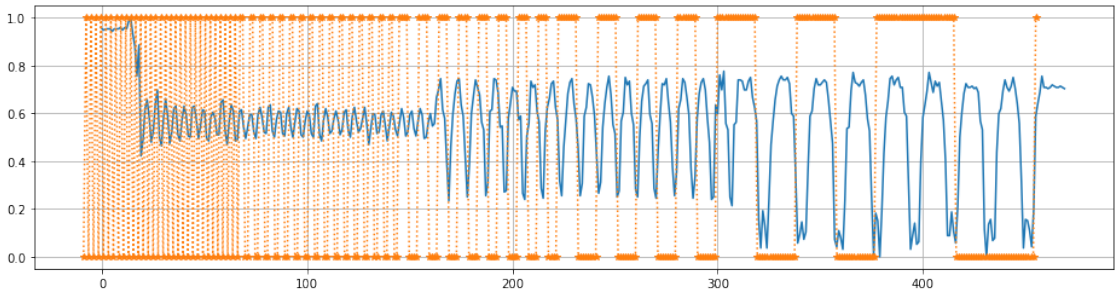
```
offset_r = 455

x_est=np.linspace(offset_l,offset_r,len(onda))
plt.plot(x_est, onda, ':*')
```

[13]: [<matplotlib.lines.Line2D at 0x7fb151e02dc0>]



[14]:
```
# idem normalizado
%matplotlib inline
plt.figure(figsize=(16,4))
plt.grid()
plt.plot((y_est_r-np.min(y_est_r))/(np.max(y_est_r)-np.min(y_est_r)))
x_est=np.linspace(offset_l,offset_r,len(onda))
plt.plot(x_est, (onda-np.min(onda))/(np.max(onda)-np.min(onda)), ':*')
```

[14]: [<matplotlib.lines.Line2D at 0x7fb151dfcac0>]



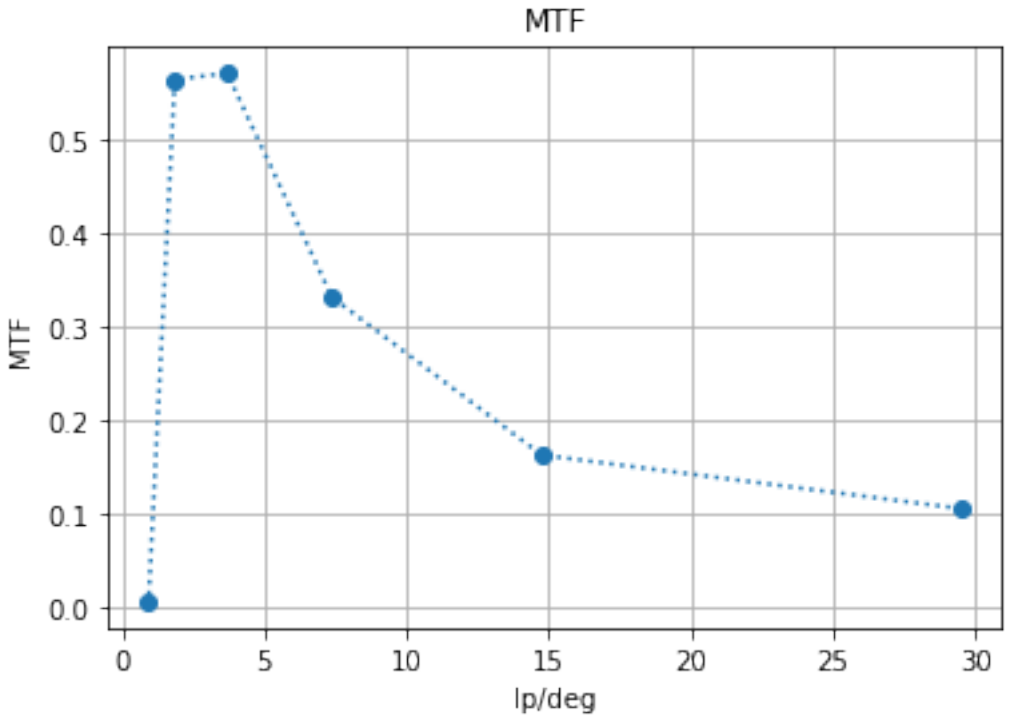
[15]:
```
# Sacar fotos y tomar nota de la geometría de cómo tomaron la foto en␣
 ↪particular distancia al monitor
# y ángulo subtendido por zona usada para calcular mtf:
distancia_monitor_mm = 500
ancho_zona_mtf_mm = 70
```

```
# Calcular ángulo que genera la zona usada para medir mtf (cateto menor␣
 ↪en la pantalla
# y cateto mayor distancia entre cámara y pantalla)
angulo_zona_mtf_deg = np.arctan2(ancho_zona_mtf_mm,␣
 ↪distancia_monitor_mm)*180/np.pi
ancho_zona_mtf_px = len(y_est_r)
```

[16]:
```
# Calculamos grados por pixel para usar en el gráfico
deg_per_px = ancho_zona_mtf_px/angulo_zona_mtf_deg
```

[17]:
```
# medimos el contraste en cada tramo y lo graficamos
tramos = np.split(y_est_r, bordes)

mtfs = []

for tramo in tramos:
    i_max = np.max(tramo).astype(np.float32)
    i_min = np.min(tramo).astype(np.float32)
    mtf = (i_max - i_min) / (i_max + i_min)
    mtfs.append(mtf)
    print(mtf)

plt.grid(); plt.title('MTF')
plt.plot(1/np.array(anchos)*deg_per_px, mtfs, ':o')
plt.ylabel('MTF');
plt.xlabel('lp/deg');
```

```
0.10467706
0.16207951
0.33112583
0.57088125
0.5642023
0.0052083335
```


MTF
0.5
0.4
0.3
0.2
0.1
0.0
MTF
0
5
10
15
20
25
30
lp/deg

[ ]: