ESEIAAT

Treball Final de Grau

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Estudi de la millora de prestacions de càmeres compactes amb el CHDK

Annex I

Alumne:
Arnau De Dios Diaz

Director:
Jordi Sellarès González

22 de juny del 2021

Índex Annex I

1 Lectura 2
2 Panoramica 5
2.1 Scripts JPG 5
2.2 Script DNG 5
2.3 Python 6
3 Bomba de neutrons 9
3.1 Scripts JPG 9
3.2 Script DNG 9
3.3 Python 10
4 Visió Nocturna 14
4.1 Scripts JPG 14
4.2 Script DNG 15
4.3 Python 17
5 Bracketing 20
5.1 Scripts JPG 20
5.2 Script DNG 21
5.3 Python 22
6 Timelapse 25
6.1 Scripts JPG 25
6.2 Script DNG 25
6.3 Python 26
7 Mode nit 28
7.1 Scripts JPG 28
7.2 Script DNG 29
7.3 Python 30
8 Pinhole 32
8.1 Scripts JPG 32
8.2 Script DNG 32
8.3 Python 33

1 Lectura

```
#-----------------------------------------------------#
#   Lectura de fitxers directament des de targeta
#   Autor: Arnau De Dios
#   Data creació: Març 2021
#-----------------------------------------------------#

"""
Lectura de fitxer directament des de targeta.

Aquest programa llegeix l'arxiu dades.log que troba al disposi
tiu E:\ del PC. D'aquí extreu dues coses, la primera es la fun
ció que ha de realitzar i el filtre que ha d'aplicar
i per a un altre lloc les imatges que participen en el procés.


Després fa una cerca de les imatges per la tarjeta. Com no tot
es les càmeres tenen els mateixos directoris el sistema es uni
versal. Únicament troba les dreceres, no carrega cap
arxiu ni obre cap imatge. Crea una nova llista amb les adreces
 senceres de les imatges, que posteriorment pasa a la funció c
orresponent per que operi convenientment.

"""

#Llibreries
import os
from tqdm import tqdm #Llibreria per poder posar una barra de
procès
from timeit import default_timer #Llibreria per calcular el te
mps de processament

from panoramica import * #Importar la funció Panoramica dintre
 del arxiu panoramica.py
from bomba import * #Importar la funció Bomba dintre del arxiu
 bomba.py
from visio import * #Importar la funció Visió dintre de l'arxi
u visio.py
from bracketing import * #Importar la funció Bracketing dintre
 de l'arxiu bracketing.py
from timelapse import * #Importat la funció Timelapse dintre d
e l'arxiu timelapse.py
from nit import * #Importat la funció Nit dintre de l'arxiu ni
t.py
from pinhole import *
#Importat la funció Pinhole dintre de l'arxiu pinhole.py

#Valors inicials
```

```
inici = default_timer() #Inicialització del temps

#Funcions
#-------#

#Lectura del fitxer Log
arxius = []
with open("E:\dades.log") as fname: #Especifiquem la ruta del
fitxer Log
    for arxiu in fname:
        arxius.extend(arxiu.split()) #Afegim a una llista els
valors de l'arxiu

#Cerca de les imatges en la tarjeta
imatges = []
for i in tqdm(range(1, len(arxius)), desc="Buscant les imatges
"): #Fem la cerca per cada un de les fotografies a processar
    dirInit = 'E:' #Donem com a paràmetre de cerca únic el dis
c on es troba la fotografia
    path = ''
    for root, _, files in os.walk(dirInit): #Cerquem dintre de
l root fent una llista de tots els fitxers
        if arxius[i] in files: #Si troba un arxiu que concidei
xi
            imatges.append(os.path.join(root, arxius[i])) #afe
geix a una llista la ruta sencera de la imatge per que poderla
 processar
            break #Com es tracta d'arxius únics trenquem la ce
rca una vegada s'ha trobat per estalviar recursos

#Procesament dels filtre
if arxius[0] == "Panoramic":
    print("Processant Panoràmica")
    panoramica(imatges)

elif arxius[0] == "Bomba":
    print("Processant Bomba de Neutrons")
    bomba(imatges)

elif arxius[0] == "Visio":
    print("Processant Visió Nocturna")
    visio(imatges)

elif arxius[0] == "Bracketing":
    print("Processant Bracketing")
    bracketing(imatges)

elif arxius[0] == "Timelapse":
```

```
    print("Processant Timelapse")
    timelapse(imatges)

elif arxius[0] == "Nit":
    print("Processant Mode Nit")
    nit(imatges)

elif arxius[0] == "Fresnel":
    print("Processant Fresnel Imager")
    pinhole(imatges)


else:
    print("La paraula clau introduida no es reconeix")


final = default_timer() #Finalització del temps
print("Temps de duració: ", int(final - inici), "s") #Temps to
tal en segons
```

2 Panoramica

2.1 Script JPG

```
-- ------------------------ --
--  PANORÀMICA              --
--  Autor: Arnau De Dios    --
--  Data creació: Març 2021 --
-- ------------------------ --

--Obertura i creació d'un fitxer de dades
logfile=io.open("A/dades.log","wb")
io.output(logfile)

--Creació de la funció de registre d'aquestes dades
function log(...)
  io.write(...)
  io.write("\n")
end

--Dades inicials
total_fotos = 9

--Avís a l'usuari
log("Panoramic")
print "PANORÀMICA"
print "No tocar la càmera durant els pròxims segons"
print "El procés comença en:"
print "3"
sleep(1000)
print "2"
sleep(1000)
print "1"
sleep(1000)

--Inici de la primera imatge i registre al fitxer de dades
shoot()
log("IMG_".. string.format("%04d",get_exp_count())..".JPG")
print("Foto num: ", "1", " de ", total_fotos+1)

--Realització de les imatges en bucle i registre al fitxer de
dades
for i = 1, total_fotos do
     print "Gira la camera"
  sleep(4000)
  shoot()
  log("IMG_".. string.format("%04d",get_exp_count())..".JPG")
  print("Foto num: ", i+1, " de ", total_fotos+1)
end

--Tancament del fitxer de dades
logfile:close()
```

2.2 Script DNG

```
-- ------------------------ --
--  PANORÀMICA              --
```

```
--  Autor: Arnau De Dios     --
--  Data creació: Març 2021  --
-- ------------------------- --

--Obertura i creació d'un fitxer de dades
logfile=io.open("A/dades.log","wb")
io.output(logfile)

--Creació de la funció de registre d'aquestes dades
function log(...)
  io.write(...)
  io.write("\n")
end

--Dades inicials
total_fotos = 9

--Avís a l'usuari
log("Panoramic")
print "PANORÀMICA"
print "No tocar la càmera durant els pròxims segons"
print "El procés comença en:"
print "3"
sleep(1000)
print "2"
sleep(1000)
print "1"
sleep(1000)

--Inici de la primera imatge i registre al fitxer de dades
shoot()
log("IMG_".. string.format("%04d",get_exp_count())..".JPG")
print("Foto num: ", "1", " de ", total_fotos+1)

--Realització de les imatges en bucle i registre al fitxer de
dades
for i = 1, total_fotos do
     print "Gira la camera"
  sleep(4000)
  shoot()
  log("IMG_".. string.format("%04d",get_exp_count())..".JPG")
  print("Foto num: ", i+1, " de ", total_fotos+1)
end

--Tancament del fitxer de dades
logfile:close()
```

2.3 Python

```
#-----------------------------#
#   Filtre Panoràmica
#   Autor: Arnau De Dios
#   Data creació: Març 2021
#-----------------------------#
```

```
#Llibreries
from progress.bar import Bar #Llibreria per poder posar una ba
rra d'estat
from tqdm import tqdm #Llibreria per poder posar una barra de
procès
import cv2 #LLibreria de tractament d'imatges que em permet fe
r la panoramica
import rawpy #Llibreria per al tractament d'imatges RAW
import numpy as np #Llibreria per poder fer operacions matemàt
iques complexes
from datetime import datetime #Llibreria per poder donar el te
mps
import sys

#Valors inicials
#--------------#

#Funcions
#Funció per saber la data d'avui
def data_diaria():
    avui = str(datetime.now().day).zfill(2) + "-
" + str(datetime.now().month).zfill(2) + "-
" + str(datetime.now().year).zfill(4) + "_" + str(datetime.now
().hour).zfill(2) + "." + str(datetime.now().minute).zfill(2)
+ "." + str(datetime.now().second).zfill(2)
    return avui

#Funció panoàmica
def panoramica(arxius):
    """
    Procés de creació del filtre PANORÀMICA.

    Agafa una llista, que ve pasada per un programa extern, on
 es troben les adreces de les imatges amb les que ha de trebal
lar. A partir d'aquestes adreces carrega les
    imatges i fa el procés per aconseguir el filtre.

    """
    #Format d'imatge entrant
    auxVers = arxius[0][len(arxius[0])-
3] + arxius[0][len(arxius[0])-2] + arxius[0][len(arxius[0])-1]

    #Càrrega de les imatges
    if auxVers == "JPG":
        print("Format JPG")

        #Lectura dels arxius
        imatges = [] #Creació d'una llista amb les imatges
```

```
        for i in tqdm(range(0, len(arxius)), desc= "Creació de
 la llista d'imatges"):
            imatges.append(np.asarray(cv2.imread(arxius[i], cv
2.IMREAD_COLOR))) #Afegir a la llista d'imatges les recollides
 al fitxer LOG
        #En aquesta lectura substitueixo la lectura del PIL pe
r la de CV2 ja que al processar les imatges via CV2 es recomen
able utilitzar les eines d'aquesta llibreria

    elif auxVers == "DNG":
        print("Format DNG")

        #Lectura dels arxius
        imatges = [] #Creació d'una llista amb les imatges
        for i in tqdm(range(0, len(arxius)), desc="Afegint ima
tges a la llista"):
            with rawpy.imread(arxius[i]) as raw:
                rgb = raw.postprocess(use_camera_wb = True, no
_auto_bright = True)
            imatges.append(rgb) #Afegir a la llista d'imatges
les recollides al fitxer LOG

    #Creació de la imatge panoràmica
    bar1 = Bar('Creant panoràmica', max=5)
    stitcher = cv2.Stitcher.create(cv2.Stitcher_PANORAMA)
    bar1.next()
    estat, panoramic = stitcher.stitch(imatges)
    bar1.next()

    #Control d'errors
    if estat != cv2.Stitcher_OK:
        print("No es pot fer la panoràmica d'aquestes imatges,
 el codi d'error es: %d" % estat)
        sys.exit(-1)
    bar1.next()

    #Tranformació al sistema de color de OpenCV
    if auxVers == "DNG":
        panoramic = cv2.cvtColor(panoramic, cv2.COLOR_RGB2BGR)
    bar1.next()

    #Creació del arxius final amb la imatge compresa
    cv2.imwrite("panoramica_" + data_diaria() + ".JPG", panora
mic)
    bar1.next()
    print()

    return None
```

3 Bomba de neutrons

3.1 Script JPG

```
-- ------------------------ --
--  BOMBA DE NEUTRONS       --
--  Autor: Arnau De Dios    --
--  Data creació: Març 2021 --
-- ------------------------ --

--Obertura i creació d'un fitxer de dades
logfile=io.open("A/dades.log","wb")
io.output(logfile)

--Creació de la funció de registre d'aquestes dades
function log(...)
  io.write(...)
  io.write("\n")
end

--Dades inicials
total_fotos = 10

--Inici del programa i avís a l'usuari
log("Bomba")
print "BOMBA DE NEUTRONS"
print "No tocar la càmera durant els pròxims segons"
print "El procés comença en:"
print "3"
sleep(1000)
print "2"
sleep(1000)
print "1"
sleep(1000)
print "NO TOCAR"

--Realització de les imatges en bucle i registre al fitxer de
dades
for i = 1, total_fotos do
  shoot()
  sleep(1000)
  log("IMG_".. string.format("%04d",get_exp_count())..".JPG")
  print("Foto num: ", i, " de ", total_fotos)
end

--Tancament del fitxer de dades
logfile:close()
```

3.2 Script DNG

```
-- ------------------------ --
--  BOMBA DE NEUTRONS       --
--  Autor: Arnau De Dios    --
--  Data creació: Març 2021 --
-- ------------------------ --

--Obertura i creació d'un fitxer de dades
```

```
logfile=io.open("A/dades.log","wb")
io.output(logfile)

--Creació de la funció de registre d'aquestes dades
function log(...)
  io.write(...)
  io.write("\n")
end

--Dades inicials
total_fotos = 10

--Inici del programa i avís a l'usuari
log("Bomba")
print "BOMBA DE NEUTRONS"
print "No tocar la càmera durant els pròxims segons"
print "El procés comença en:"
print "3"
sleep(1000)
print "2"
sleep(1000)
print "1"
sleep(1000)
print "NO TOCAR"

--Realització de les imatges en bucle i registre al fitxer de
dades
for i = 1, total_fotos do
  shoot()
  sleep(1000)
  log("CRW_".. string.format("%04d",get_exp_count())..".DNG")
  print("Foto num: ", i, " de ", total_fotos)
end

--Tancament del fitxer de dades
logfile:close()
```

3.3 Python

```
#------------------------------#
#   Filtre Bomba de neutrons
#   Autors: Arnau De Dios
#   Data creació: Març 2021
#------------------------------#

#Llibreries
from datetime import datetime #Llibreria per poder donar el te
mps
import numpy as np #Llibreria per poder fer operacions matemàt
iques complexes
from PIL import Image #Llibreria per poder carregar les imatge
s
import rawpy #Llibreria per al tractament d'imatges RAW
```

```
from tqdm import tqdm #Llibreria per poder posar una barra d'e
stat
from tqdm import trange #Llibreria per poder posar una barra d
'estat

#Valors inicials
#--------------#

#Funcions
#Funció inliers
def inliers(x, alpha = 1.0):
    sd = np.std(x)
    xm = np.average(x)
    while True:
        sortida = True
        for i, y in enumerate(x):
            if np.abs(y - xm) > alpha*sd:
                x = np.concatenate((x[:i], x[i+1:]))
                sortida = False
                break
        if sortida == True:
            return np.average(x)

#Funció per saber la data d'avui
def data_diaria():
    avui = str(datetime.now().day).zfill(2) + "-
" + str(datetime.now().month).zfill(2) + "-
" + str(datetime.now().year).zfill(4) + "_" + str(datetime.now
().hour).zfill(2) + "." + str(datetime.now().minute).zfill(2)
+ "." + str(datetime.now().second).zfill(2)
    return avui

#Funció Bomba
def bomba(arxius):
    """
    Procés de creació del filtre BOMBA DE NEUTRONS.

    Agafa una llista, que ve pasada per un programa extern, on
 es troben les adreces de les imatges amb les que ha de trebal
lar. A partir d'aquestes adreces carrega les
    imatges i fa el procés per aconseguir el filtre.

    """

    #Format d'imatge entrant
    auxVers = arxius[0][len(arxius[0])-
3] + arxius[0][len(arxius[0])-2] + arxius[0][len(arxius[0])-1]
```

```
    #Càrrega de les imatges
    if auxVers == "JPG":
        print("Format JPG")

        #Lectura dels arxius
        images = [] #Creació d'una llista amb les imatges
        for i in tqdm(range(0, len(arxius)), desc= "Creació de
 la llista d'imatges"):
            images.append(np.asarray(Image.open(arxius[i])))
      #En aquesta lectura substitueixo la lectura del PIL per
la de CV2 ja que al processar les imatges via CV2 es recomenab
le utilitzar les eines d'aquesta llibreria

    elif auxVers == "DNG":
        print("Format DNG")

        #Lectura dels arxius
        images = [] #Creació d'una llista amb les imatges
        for i in tqdm(range(0, len(arxius)), desc="Afegint ima
tges a la llista"):
            with rawpy.imread(arxius[i]) as raw:
                rgb = raw.postprocess(use_camera_wb = True)
            images.append(rgb) #Afegir a la llista d'imatges l
es recollides al fitxer LOG


    #Valors inicials
    coordY, coordX, color = images[0].shape
    print(images[0].shape)

    #Creació de la imatge de sortida
    final_image =  Image.new('RGB', (coordX,coordY))

    #Separació de les imatges en pixels
    values = []
    for x in tqdm(range(coordX), desc="Separació dels pixels")
:
        column_values = []
        for y in range(coordY):
            R_values = []
            G_values = []
            B_values = []
            for i in images:
                R = i[y][x][0]
                G = i[y][x][1]
                B = i[y][x][2]
                R_values.append(R)
                G_values.append(G)
```

```
                B_values.append(B)
            column_values.append([R_values, G_values, B_values
])
        values.append(column_values)

    #Creació de la imatge amb la funció inliers
    for y in tqdm(range(coordY), desc="Creant la imatge result
ant"):
        for x in range(coordX):
            R_values = values[x][y][0]
            G_values = values[x][y][1]
            B_values = values[x][y][2]
            R = int(inliers(R_values))
            G = int(inliers(G_values))
            B = int(inliers(B_values))
            final_image.putpixel((x,y), (R,G,B))
    final_image.save("bomba_" + data_diaria() + ".jpg") #Guard
em el resultat en una nova imatge

    return None
```

4 Visió Nocturna

4.1 Script JPG

```
-- ------------------------- --
--  VISIÓ NOCTURNA           --
--  Autor: Arnau De Dios     --
--  Data creació: Abril 2021 --
-- ------------------------- --

--Obertura i creació d'un fitxer de dades
logfile=io.open("A/dades.log","wb")
io.output(logfile)

--Creació de la funció de registre d'aquestes dades
function log(...)
  io.write(...)
  io.write("\n")
end

--Dades inicials
total_fotos = 10
tipus_prop = get_propset()

val_iso = 20000 --La ISO s'ha d'afegir un 0 extra (8000 de
valor es 800 ISO)
val_flash = 2 --Flash deshabilitat
val_shutter = 20000000 --Velocitat d'obturació a 1 segon per
defecte
val_display = 2 --Display deshabilitat
val_display_light = 0 --Llum display deshabilitat


--Establiment dels valors segons propset (Tipus de processador
DIGIC incorporat)
if tipus_prop == 1 then
     set_prop(16,val_flash) --Deshabilitar el flash
     set_prop(21,val_iso) --Fixar la ISO
     set_tv96(usec_to_tv96(val_shutter)) --Fixar la velocitat

     set_backlight(val_display_light) --Fixar la llum de la
pantalla apagada
     set_lcd_display(val_display) --Fixar la pantalla LCD
apagada

else --La resta de processadors (o propsets) utilitzen els
mateixos llocs de memòria
     set_prop(143,val_flash) --Deshabilitar el flash
     set_prop(149,val_iso) --Fixar la ISO
     set_tv96(usec_to_tv96(val_shutter)) --Fixar la velocitat
     set_backlight(val_display_light) -- Fixar la llum de la
pantalla apagada
     set_lcd_display(val_display) -- Fixar la pantalla LCD
apagada
end

--Inici del programa i avís a l'usuari
```

```
log("Visio")
print "VISIÓ NOCTURNA"
print "No tocar la càmera durant els pròxims segons"
print "El procés comença en:"
print "3"
sleep(1000)
print "2"
sleep(1000)
print "1"
sleep(1000)
print "NO TOCAR"

--Realització de les imatges en bucle i registre al fitxer de
dades
for i = 1, total_fotos do
     set_tv96(usec_to_tv96(val_shutter)) --Els set de funció
interna de CHDK s'han d'actualitzar a totes les iteracions
     set_backlight(0)
     set_lcd_display(0)
     shoot()
     log("IMG_"..
string.format("%04d",get_exp_count())..".JPG")
     print("Foto num: ", i, " de ", total_fotos)
end

--Tancament del fitxer de dades
logfile:close()
```

4.2 Script DNG

```
-- -------------------------- --
--  VISIÓ NOCTURNA            --
--  Autor: Arnau De Dios      --
--  Data creació: Abril 2021  --
-- -------------------------- --

--Obertura i creació d'un fitxer de dades
logfile=io.open("A/dades.log","wb")
io.output(logfile)

--Creació de la funció de registre d'aquestes dades
function log(...)
  io.write(...)
  io.write("\n")
end

--Dades inicials
total_fotos = 10
tipus_prop = get_propset()

val_iso = 20000 --La ISO s'ha d'afegir un 0 extra (8000 de
valor es 800 ISO)
val_flash = 2 --Flash deshabilitat
val_shutter = 20000000 --Velocitat d'obturació a 1 segon per
defecte
val_display = 2 --Display deshabilitat
```

```
val_display_light = 0 --Llum display deshabilitat


--Establiment dels valors segons propset (Tipus de processador
DIGIC incorporat)
if tipus_prop == 1 then
     set_prop(16,val_flash) --Deshabilitar el flash
     set_prop(21,val_iso) --Fixar la ISO
     set_tv96(usec_to_tv96(val_shutter)) --Fixar la velocitat

     set_backlight(val_display_light) --Fixar la llum de la
pantalla apagada
     set_lcd_display(val_display) --Fixar la pantalla LCD
apagada

else --La resta de processadors (o propsets) utilitzen els
mateixos llocs de memòria
     set_prop(143,val_flash) --Deshabilitar el flash
     set_prop(149,val_iso) --Fixar la ISO
     set_tv96(usec_to_tv96(val_shutter)) --Fixar la velocitat
     set_backlight(val_display_light) -- Fixar la llum de la
pantalla apagada
     set_lcd_display(val_display) -- Fixar la pantalla LCD
apagada
end

--Inici del programa i avís a l'usuari
log("Visio")
print "VISIÓ NOCTURNA"
print "No tocar la càmera durant els pròxims segons"
print "El procés comença en:"
print "3"
sleep(1000)
print "2"
sleep(1000)
print "1"
sleep(1000)
print "NO TOCAR"

--Realització de les imatges en bucle i registre al fitxer de
dades
for i = 1, total_fotos do
     set_tv96(usec_to_tv96(val_shutter)) --Els set de funció
interna de CHDK s'han d'actualitzar a totes les iteracions
     set_backlight(0)
     set_lcd_display(0)
     shoot()
     log("CRW_"..
string.format("%04d",get_exp_count())..".DNG")
     print("Foto num: ", i, " de ", total_fotos)
end

--Tancament del fitxer de dades
logfile:close()
```

4.3 Python

```
#-----------------------------#
#   Filtre Visió Nocturna
#   Autor: Arnau De Dios
#   Data creació: Març 2021
#-----------------------------#

#Llibreries
from datetime import datetime #Llibreria per poder donar el te
mps
import numpy as np #Llibreria per poder fer operacions matemàt
iques complexes
from PIL import Image #Llibreria per poder carregar les imatge
s
from numpy.lib.function_base import append
from numpy.lib.type_check import imag #Llibreria per al tracta
ment d'imatges
import rawpy #Llibreria per al tractament d'imatges RAW
import cv2 #LLibreria de tractament d'imatges que em permet fe
r la panoramica
from tqdm import tqdm #Llibreria per poder posar una barra d'e
stat

#Valors inicials
#--------------#

#Funcions
#Funció per saber la data d'avui
def data_diaria():
    avui = str(datetime.now().day).zfill(2) + "-
" + str(datetime.now().month).zfill(2) + "-
" + str(datetime.now().year).zfill(4) + "_" + str(datetime.now
().hour).zfill(2) + "." + str(datetime.now().minute).zfill(2)
+ "." + str(datetime.now().second).zfill(2)
    return avui

#Funció per normalitzar una imatge
def normalitzar(entrada):
    imgAux = np.array(entrada)
    R, G, B = cv2.split(imgAux)

    output1_R = cv2.equalizeHist(R)
    output1_G = cv2.equalizeHist(G)
    output1_B = cv2.equalizeHist(B)

    sortida = cv2.merge((output1_R, output1_G, output1_B))

    return sortida
```

```
#Funció Visió nocturna
def visio(arxius):
    """
    Procés de creació del filtre VISIÓ NOCTURNA

    Agafa una llista, que ve pasada per un programa extern, on
 es troben les adreces de les imatges amb les que ha de trebal
lar. A partir d'aquestes adreces carrega les
    imatges i fa el procés per aconseguir el filtre.

    """

    #Format d'imatge entrant
    auxVers = arxius[0][len(arxius[0])-
3] + arxius[0][len(arxius[0])-2] + arxius[0][len(arxius[0])-1]

    #Càrrega de les imatge
    if auxVers == "JPG":
        print("Format JPG")

        #Lectura dels arxius
        images = [] #Creació d'una llista amb les imatges
        for i in tqdm(range(0, len(arxius)), desc="Afegint ima
tges a la llista"):
            images.append(np.asarray(Image.open(arxius[i]))) #
Afegir a la llista d'imatges les recollides al fitxer LOG

    elif auxVers == "DNG":
        print("Format DNG")

        #Lectura dels arxius
        images = [] #Creació d'una llista amb les imatges
        for i in tqdm(range(0, len(arxius)), desc="Afegint ima
tges a la llista"):
            with rawpy.imread(arxius[i]) as raw:
                rgb = raw.postprocess(use_camera_wb = True, no
_auto_bright = True)
                images.append(rgb) #Afegir a la llista d'imatg
es les recollides al fitxer LOG


    #Valors inicials
    coordY, coordX, color = np.array(images[0]).shape

    #Creació de la imatge final
    final_image = Image.new('RGB', (coordX,coordY))

    #Creació de la imatge resultant
```

```
    values = []
    for x in tqdm(range(coordX), desc="Creació de la imatge re
sultant"):
        column_values = []
        for y in range(coordY):
            R_values = 0
            G_values = 0
            B_values = 0
            for i in images:
                R = i[y][x][0]
                G = i[y][x][1]
                B = i[y][x][2]
                R_values = R_values + R
                G_values = G_values + G
                B_values = B_values + B
            final_image.putpixel((x,y), (R_values,G_values,B_v
alues))
            column_values.append([R_values, G_values, B_values
])
        values.append(column_values)


    #Cració de la imatge normalitzada
    norm_image = normalitzar(final_image)

    #Separació en components
    R, G, B = cv2.split(norm_image)

    #Denoising del canal vermell
    deno_vermell = cv2.fastNlMeansDenoising(R,None,100,7,21)

    #Guardo la imatge
    cv2.imwrite("visio_" + data_diaria() + ".JPG", deno_vermel
l)

    return None
```

5 Bracketing

5.1 Script JPG

```
-- -------------------------- --
--  BRACKETING                --
--  Autor: Arnau De Dios      --
--  Data creació: Maig 2021   --
-- -------------------------- --

--Obertura i creació d'un fitxer de dades
logfile=io.open("A/dades.log","wb")
io.output(logfile)

--Creació de la funció de registre d'aquestes dades
function log(...)
  io.write(...)
  io.write("\n")
end

--Dades inicials
total_fotos = 13
tipus_prop = get_propset()

val_flash = 2 --Flash deshabilitat
val_exp = -200 --Exposura inicial

--Establiment dels valors segons propset (Tipus de processador
DIGIC incorporat)
if tipus_prop == 1 then
     set_prop(16,val_flash) --Deshabilitar el flash
else --La resta de processadors (o propsets) utilitzen els
mateixos llocs de memòria
     set_prop(143,val_flash) --Deshabilitar el flash
end

--Inici del programa i avís a l'usuari
log("Bracketing")
print "BRACKETING"
print "No tocar la càmera durant els pròxims segons"
print "El procés comença en:"
print "3"
sleep(1000)
print "2"
sleep(1000)
print "1"
sleep(1000)
print "NO TOCAR"

--Realització de les imatges en bucle i registre al fitxer de
dades
for i = 1, total_fotos do
     set_ev(val_exp)
     shoot()
     log("IMG_"..
string.format("%04d",get_exp_count())..".JPG")
     print("Foto num: ", i, " de ", total_fotos)
```

```
     val_exp = val_exp + 40
end

--Tancament del fitxer de dades
logfile:close()
```

5.2 Script DNG

```
-- -------------------------- --
--  BRACKETING                --
--  Autor: Arnau De Dios      --
--  Data creació: Maig 2021   --
-- -------------------------- --

--Obertura i creació d'un fitxer de dades
logfile=io.open("A/dades.log","wb")
io.output(logfile)

--Creació de la funció de registre d'aquestes dades
function log(...)
  io.write(...)
  io.write("\n")
end

--Dades inicials
total_fotos = 13
tipus_prop = get_propset()

val_flash = 2 --Flash deshabilitat
val_exp = -200 --Exposura inicial

--Establiment dels valors segons propset (Tipus de processador
DIGIC incorporat)
if tipus_prop == 1 then
     set_prop(16,val_flash) --Deshabilitar el flash
else --La resta de processadors (o propsets) utilitzen els
mateixos llocs de memòria
     set_prop(143,val_flash) --Deshabilitar el flash
end

--Inici del programa i avís a l'usuari
log("Bracketing")
print "BRACKETING"
print "No tocar la càmera durant els pròxims segons"
print "El procés comença en:"
print "3"
sleep(1000)
print "2"
sleep(1000)
print "1"
sleep(1000)
print "NO TOCAR"

--Realització de les imatges en bucle i registre al fitxer de
dades
for i = 1, total_fotos do
```

```
      set_ev(val_exp)
      shoot()
      log("CRW_"..
string.format("%04d",get_exp_count())..".DNG")
      print("Foto num: ", i, " de ", total_fotos)
      val_exp = val_exp + 40
end

--Tancament del fitxer de dades
logfile:close()
```

5.3 Python

```
#-----------------------------#
#   Filtre Bracketing
#   Autor: Arnau De Dios
#   Data creació: Maig 2021
#-----------------------------#

#Llibreries
from progress.bar import Bar #Llibreria per poder posar una ba
rra d'estat
from tqdm import tqdm #Llibreria per poder posar una barra de
procès
import cv2 #LLibreria de tractament d'imatges que em permet fe
r la panoramica
import rawpy.enhance #Llibreria per al tractament d'imatges RA
W
import numpy as np #Llibreria per poder fer operacions matemàt
iques complexes
from datetime import datetime #Llibreria per poder donar el te
mps

#Valors inicials
#-------------#

#Funcions
#Funció per saber la data d'avui
def data_diaria():
    avui = str(datetime.now().day).zfill(2) + "-
" + str(datetime.now().month).zfill(2) + "-
" + str(datetime.now().year).zfill(4) + "_" + str(datetime.now
().hour).zfill(2) + "." + str(datetime.now().minute).zfill(2)
+ "." + str(datetime.now().second).zfill(2)
    return avui

#Funció bracketing
def bracketing(arxius):
    """
    Procés de creació del filtre BRACKETING.
```

```
    Agafa una llista, que ve pasada per un programa extern, on
 es troben les adreces de les imatges amb les que ha de trebal
lar. A partir d'aquestes adreces carrega les
    imatges i fa el procés per aconseguir el filtre.

    """

    #Format imatge d'entrada
    auxVers = arxius[0][len(arxius[0])-
3] + arxius[0][len(arxius[0])-2] + arxius[0][len(arxius[0])-1]

    #Càrrega de les imatges
    if auxVers == "JPG":
        print("Format JPG")

        #Lectura dels arxius
        imatges = [] #Creació d'una llista amb les imatges
        for i in tqdm(range(0, len(arxius)), desc= "Creació de
 la llista d'imatges"):
            imatges.append(np.asarray(cv2.imread(arxius[i], cv
2.IMREAD_COLOR))) #Afegir a la llista d'imatges les recollides
 al fitxer LOG
        #En aquesta lectura substitueixo la lectura del PIL pe
r la de CV2 ja que al processar les imatges via CV2 es recomen
able utilitzar les eines d'aquesta llibreria

    elif auxVers == "DNG":
        print("Format DNG")

        #Lectura dels arxius
        imatges = [] #Creació d'una llista amb les imatges
        for i in tqdm(range(0, len(arxius)), desc="Afegint ima
tges a la llista"):
            with rawpy.imread(arxius[i]) as raw:
                rgb = raw.postprocess(use_camera_wb = True, no
_auto_bright = True)
            imatges.append(rgb) #Afegir a la llista d'imatges
les recollides al fitxer LOG

    #Creació de la barra d'estat
    bar1 = Bar('Creant la imatge fusionada:', max=6)

    #Creació d'alineador d'imatges
    alinear_imatges = cv2.createAlignMTB()
    bar1.next()

    #Alinear les imatges
```

```
    alinear_imatges.process(imatges, imatges)
    bar1.next()

    #Creació del juntador d'imatges
    mergeMertens = cv2.createMergeMertens()
    bar1.next()

    #Juntar les matges
    imatge_fusionada = mergeMertens.process(imatges)
    bar1.next()

    if auxVers == "DNG":
        imatge_fusionada = cv2.cvtColor(imatge_fusionada, cv2.
COLOR_RGB2BGR)
    bar1.next()

    #Creació de la imatge final
    cv2.imwrite("bracketing_" + data_diaria() + ".JPG", imatge
_fusionada * 255)
    bar1.next()
    print()

    return None
```

6 Timelapse

6.1 Script JPG

```
-- ------------------------ --
--  TIMELAPSE               --
--  Autor: Arnau De Dios    --
--  Data creació: Maig 2021 --
-- ------------------------ --

--Obertura i creació d'un fitxer de dades
logfile=io.open("A/dades.log","wb")
io.output(logfile)

--Creació de la funció de registre d'aquestes dades
function log(...)
  io.write(...)
  io.write("\n")
end

--Dades inicials
total_fotos = 80

--Inici del programa i avís a l'usuari
log("Timelapse")
print "Timelapse"
print "No tocar la càmera durant els pròxims segons"
print "El procés comença en:"
print "3"
sleep(1000)
print "2"
sleep(1000)
print "1"
sleep(1000)
print "NO TOCAR"

--Realització de les imatges en bucle i registre al fitxer de
dades
for i = 1, total_fotos do
  shoot()
  sleep(3750)
  log("IMG_".. string.format("%04d",get_exp_count())..".JPG")
  print("Foto num: ", i, " de ", total_fotos)
end

--Tancament del fitxer de dades
logfile:close()
```

6.2 Script DNG

```
-- ------------------------ --
--  TIMELAPSE               --
--  Autor: Arnau De Dios    --
--  Data creació: Maig 2021 --
-- ------------------------ --

--Obertura i creació d'un fitxer de dades
```

```
logfile=io.open("A/dades.log","wb")
io.output(logfile)

--Creació de la funció de registre d'aquestes dades
function log(...)
  io.write(...)
  io.write("\n")
end

--Dades inicials
total_fotos = 80

--Inici del programa i avís a l'usuari
log("Timelapse")
print "Timelapse"
print "No tocar la càmera durant els pròxims segons"
print "El procés comença en:"
print "3"
sleep(1000)
print "2"
sleep(1000)
print "1"
sleep(1000)
print "NO TOCAR"

--Realització de les imatges en bucle i registre al fitxer de
dades
for i = 1, total_fotos do
  shoot()
  sleep(3750)
  log("CRW_".. string.format("%04d",get_exp_count())..".DNG")
  print("Foto num: ", i, " de ", total_fotos)
end

--Tancament del fitxer de dades
logfile:close()
```

6.3 Python

```
#-----------------------------#
#   Filtre Timelapse
#   Autor: Arnau De Dios
#   Data creació: Maig 2021
#-----------------------------#

#Llibreries
from datetime import datetime #Llibreria per poder donar el te
mps

import os

#Valors inicials
#--------------#
```

```
#Funcions
#Funció per saber la data d'avui
def data_diaria():
    avui = str(datetime.now().day).zfill(2) + "-
" + str(datetime.now().month).zfill(2) + "-
" + str(datetime.now().year).zfill(4) + "_" + str(datetime.now
().hour).zfill(2) + "." + str(datetime.now().minute).zfill(2)
+ "." + str(datetime.now().second).zfill(2)
    return avui

#Funció timelapse
def timelapse(arxius):

    #Valors inicials
    auxVers = arxius[0][len(arxius[0])-
3] + arxius[0][len(arxius[0])-2] + arxius[0][len(arxius[0])-1]
    fps = 25
    w = 1920
    h = 1080
    entrada = ''
    inicial = ''

    #Número de la primera imatge
    for i in range(len(arxius[0])-8,len(arxius[0])-4):
        inicial = inicial + arxius[0][i]

    #Arrel de les imatges
    for i in range(len(arxius[0])-9):
        if arxius[0][i] == "\\":
            entrada = entrada + '\\'
        else:
            entrada = entrada + arxius[0][i]

    #Instrucció per la creació del video
    instruccio = 'ffmpeg -hide_banner -loglevel error -
start_number ' + inicial + ' -
i ' + entrada +'_%4d.' + auxVers + ' -
framerate ' + str(fps) + ' -
s:v ' + str(w) +'x' + str(h) + ' timelapse_' + data_diaria() +
 '.mp4'

    #Trucada al sistema per realitzar el video
    os.system(instruccio)

    return None
```

7 Mode nit

7.1 Script JPG

```
-- -------------------------- --
--  MODE NIT                  --
--  Autor: Arnau De Dios      --
--  Data creació: Abril 2021  --
-- -------------------------- --

--Obertura i creació d'un fitxer de dades
logfile=io.open("A/dades.log","wb")
io.output(logfile)

--Creació de la funció de registre d'aquestes dades
function log(...)
  io.write(...)
  io.write("\n")
end

--Dades inicials
val_flash = 2 --Flash deshabilitat

--Establiment dels valors segons propset (Tipus de processador
DIGIC incorporat)
if tipus_prop == 1 then
     set_prop(16,val_flash) --Deshabilitar el flash
else --La resta de processadors (o propsets) utilitzen els
mateixos llocs de memòria
     set_prop(143,val_flash) --Deshabilitar el flash
end

--Inici del programa i avís a l'usuari
log("Nit")
print "MODE NIT"
print "No tocar la càmera durant els pròxims segons"
print "El procés comença en:"
print "3"
sleep(1000)
print "2"
sleep(1000)
print "1"
sleep(1000)
print "NO TOCAR"

--Realització de les imatges i registre al fitxer de dades
--Sense dark
set_raw_nr(1) --Fixar la NO presa del dark
shoot()
log("IMG_".. string.format("%04d",get_exp_count())..".JPG")
print("Foto num: 1 de 2")

--Amb dark
set_raw_nr(2) --Fixar la presa del dark
shoot()
log("IMG_".. string.format("%04d",get_exp_count())..".JPG")
print("Foto num: 2 de 2")
```

```
--Tancament del fitxer de dades
logfile:close()
```

7.2 Script DNG

```
-- -------------------------- --
--  MODE NIT                  --
--  Autor: Arnau De Dios      --
--  Data creació: Abril 2021  --
-- -------------------------- --

--Obertura i creació d'un fitxer de dades
logfile=io.open("A/dades.log","wb")
io.output(logfile)

--Creació de la funció de registre d'aquestes dades
function log(...)
  io.write(...)
  io.write("\n")
end

--Dades inicials
val_flash = 2 --Flash deshabilitat


--Establiment dels valors segons propset (Tipus de processador
DIGIC incorporat)
if tipus_prop == 1 then
     set_prop(16,val_flash) --Deshabilitar el flash
else --La resta de processadors (o propsets) utilitzen els
mateixos llocs de memòria
     set_prop(143,val_flash) --Deshabilitar el flash
end

--Inici del programa i avís a l'usuari
log("Nit")
print "MODE NIT"
print "No tocar la càmera durant els pròxims segons"
print "El procés comença en:"
print "3"
sleep(1000)
print "2"
sleep(1000)
print "1"
sleep(1000)
print "NO TOCAR"

--Realització de les imatges i registre al fitxer de dades
--Sense dark
set_raw_nr(1) --Fixar la NO presa del dark
shoot()
log("CRW_".. string.format("%04d",get_exp_count())..".DNG")
print("Foto num: 1 de 2")

--Amb dark
set_raw_nr(2) --Fixar la presa del dark
```

```
shoot()
log("CRW_".. string.format("%04d",get_exp_count())..".DNG")
print("Foto num: 2 de 2")

--Tancament del fitxer de dades
logfile:close()
```

7.3 Python

```
#-----------------------------#
#   Filtre Mode Nit
#   Autor: Arnau De Dios
#   Data creació: Maig 2021
#-----------------------------#

#Llibreries
from tqdm import tqdm #Llibreria per poder posar una barra de
procès
import cv2 #LLibreria de tractament d'imatges que em permet fe
r la panoramica
import rawpy #Llibreria per al tractament d'imatges RAW
import numpy as np #Llibreria per poder fer operacions matemàt
iques complexes
from datetime import datetime #Llibreria per poder donar el te
mps

#Valors inicials
#--------------#

#Funcions
#Funció per saber la data d'avui
def data_diaria():
    avui = str(datetime.now().day).zfill(2) + "-
" + str(datetime.now().month).zfill(2) + "-
" + str(datetime.now().year).zfill(4) + "_" + str(datetime.now
().hour).zfill(2) + "." + str(datetime.now().minute).zfill(2)
+ "." + str(datetime.now().second).zfill(2)
    return avui

#Funció per calcular la realció senyal soroll
def snr(entrada, axis = 0, ddof = 0):
    entrada = np.asanyarray(entrada)
    m = entrada.mean(axis)
    sd = entrada.std(axis = axis, ddof = ddof)
    sortida = np.where(sd == 0, 0, m/sd)
    return np.mean(sortida)

#Funció Nit
def nit(arxius):
    """
```

```
    Procés de creació del filtre MODE NIT.

    Agafa una llista, que ve pasada per un programa extern, on
 es troben les adreces de les imatges amb les que ha de trebal
lar. A partir d'aquestes adreces carrega les
    imatges i fa el procés per aconseguir el filtre.

    """
    #Format d'imatges entrants
    auxVers = arxius[0][len(arxius[0])-
3] + arxius[0][len(arxius[0])-2] + arxius[0][len(arxius[0])-1]

    #Càrrega de les imatges
    if auxVers == "JPG":
        print("Format JPG")

        #Lectura dels arxius
        imatges = [] #Creació d'una llista amb les imatges
        for i in tqdm(range(0, len(arxius)), desc= "Creació de
 la llista d'imatges"):
            imatges.append(np.asarray(cv2.imread(arxius[i], cv
2.IMREAD_COLOR))) #Afegir a la llista d'imatges les recollides
 al fitxer LOG
        #En aquesta lectura substitueixo la lectura del PIL pe
r la de CV2 ja que al processar les imatges via CV2 es recomen
able utilitzar les eines d'aquesta llibreria

    elif auxVers == "DNG":
        print("Format DNG")

        #Lectura dels arxius
        imatges = [] #Creació d'una llista amb les imatges
        for i in tqdm(range(0, len(arxius)), desc="Afegint ima
tges a la llista"):
            with rawpy.imread(arxius[i]) as raw:
                rgb = raw.postprocess(use_camera_wb = True, no
_auto_bright = True)
            imatges.append(rgb) #Afegir a la llista d'imatges
les recollides al fitxer LOG

    #Càlcul del valor de la funció snr
    for i in range(len(imatges)):
        valor_snr = snr(imatges[i])
        print("La relació senyal-
soroll de la imatge es: ", valor_snr)

    return None
```

8 Pinhole

8.1 Script JPG

```
-- -------------------------- --
--  PINHOLE                   --
--  Autor: Arnau De Dios      --
--  Data creació: Abril 2021  --
-- -------------------------- --

--Obertura i creació d'un fitxer de dades
logfile=io.open("A/dades.log","wb")
io.output(logfile)

--Creació de la funció de registre d'aquestes dades
function log(...)
  io.write(...)
  io.write("\n")
end

--Inici del programa i avís a l'usuari
log("Pinhole")
print "Pinhole"
print "No tocar la càmera durant els pròxims segons"
print "El procès començarà en:"
print "3"
sleep(1000)
print "2"
sleep(1000)
print "1"
sleep(1000)
print "NO TOCAR"

--Realització de les imatges en bucle i registre al fitxer de
dades
shoot()
log("IMG_".. string.format("%04d",get_exp_count())..".JPG")

--Tancament del fitxer de dades
logfile:close()
```

8.2 Script DNG

```
-- -------------------------- --
--  PINHOLE                   --
--  Autor: Arnau De Dios      --
--  Data creació: Abril 2021  --
-- -------------------------- --

--Obertura i creació d'un fitxer de dades
logfile=io.open("A/dades.log","wb")
io.output(logfile)

--Creació de la funció de registre d'aquestes dades
function log(...)
  io.write(...)
  io.write("\n")
```

```
end

--Inici del programa i avís a l'usuari
log("Pinhole")
print "Pinhole"
print "No tocar la càmera durant els pròxims segons"
print "El procès començarà en:"
print "3"
sleep(1000)
print "2"
sleep(1000)
print "1"
sleep(1000)
print "NO TOCAR"

--Realització de les imatges en bucle i registre al fitxer de
dades
shoot()
log("CRW_".. string.format("%04d",get_exp_count())..".DNG")

--Tancament del fitxer de dades
logfile:close()
```

8.3 Python

```
#-----------------------------#
#   Filtre Pinhole
#   Autor: Arnau De Dios
#   Data creació: Maig 2021
#-----------------------------#

#Llibreries
from progress.bar import Bar #Llibreria per poder posar una ba
rra d'estat
from tqdm import tqdm #Llibreria per poder posar una barra de
procès
import cv2 #LLibreria de tractament d'imatges que em permet fe
r la panoramica
import rawpy #Llibreria per al tractament d'imatges RAW
import numpy as np
from PIL import Image #Llibreria per poder carregar les imatge
s
import matplotlib.pyplot as plt
import scipy
from scipy.signal import convolve
from scipy import ndimage
from timeit import default_timer #Llibreria per calcular el te
mps de processament
from datetime import datetime #Llibreria per poder donar el te
mps
import sys
```

```
#Valors inicials
#--------------#

#Funcions
#Funció per saber la data d'avui
def data_diaria():
    avui = str(datetime.now().day).zfill(2) + "-
" + str(datetime.now().month).zfill(2) + "-
" + str(datetime.now().year).zfill(4) + "_" + str(datetime.now
().hour).zfill(2) + "." + str(datetime.now().minute).zfill(2)
+ "." + str(datetime.now().second).zfill(2)
    return avui

#Funció Pinhole
def pinhole(arxius):
    """
    Procés de creació del filtre PINHOLE.

    Agafa una llista, que ve pasada per un programa extern, on
 es troben les adreces de les imatges amb les que ha de trebal
lar. A partir d'aquestes adreces carrega les
    imatges i fa el procés per aconseguir el filtre.

    """

    #Format d'imatge entrant
    auxVers = arxius[0][len(arxius[0])-
3] + arxius[0][len(arxius[0])-2] + arxius[0][len(arxius[0])-1]

    #Càrrega de les imatges
    if auxVers == "JPG":
        print("Format JPG")

        #Lectura dels arxius
        imatges = [] #Creació d'una llista amb les imatges
        for i in tqdm(range(0, len(arxius)), desc= "Creació de
 la llista d'imatges"):
            imatges.append(np.asarray(cv2.imread(arxius[i], cv
2.IMREAD_COLOR))) #Afegir a la llista d'imatges les recollides
 al fitxer LOG
        #En aquesta lectura substitueixo la lectura del PIL pe
r la de CV2 ja que al processar les imatges via CV2 es recomen
able utilitzar les eines d'aquesta llibreria

    elif auxVers == "DNG":
        print("Format DNG")

        #Lectura dels arxius
```

```
        imatges = [] #Creació d'una llista amb les imatges
        for i in tqdm(range(0, len(arxius)), desc="Afegint ima
tges a la llista"):
            with rawpy.imread(arxius[i]) as raw:
                rgb = raw.postprocess(use_camera_wb = True, no
_auto_bright = True)
            imatges.append(rgb) #Afegir a la llista d'imatges
les recollides al fitxer LOG

    #Creació del cercle
    bar1 = Bar('Creant el cercle', max=7)
    tamany = 24
    bar1.next()

    imatge_inicial = Image.new('RGB', (int(tamany), int(tamany
)), color='black')
    bar1.next()

    center_coordinates = (int(tamany/2), int(tamany/2))
    bar1.next()

    radius = int(tamany/2)
    bar1.next()

    color = (255, 255, 255)
    bar1.next()

    thickness = -1
    bar1.next()

    kernel = cv2.circle(np.array(imatge_inicial), center_coord
inates, radius, color, thickness)
    bar1.next()

    kernel = np.asarray(kernel)

    print() #Per que no es juntin els bars

    #Separació per componets de color
    bar2 = Bar('Separació per components de color', max=2)
    R, G, B = cv2.split(imatges[0])
    bar2.next()
    k1, k2, k3 = cv2.split(np.float32(kernel))
    bar2.next()

    print() #Per que no es juntin els bars

    #Convolució per cada un dels canals de color
```

```
    bar3 = Bar('Convolució per canals de color', max=4)
    conv_R = ndimage.convolve(R, np.asarray(k1))
    bar3.next()
    conv_G = ndimage.convolve(G, np.asarray(k2))
    bar3.next()
    conv_B = ndimage.convolve(B, np.asarray(k3))
    bar3.next()
    imatge_conv = cv2.merge((conv_R , conv_G , conv_B ))
    bar3.next()

    print() #Per que no es juntin els bars

    #Creació de la imatge final
    bar4 = Bar('Convolució per canals de color', max=3)
    imatge_norm = cv2.normalize(imatge_conv, None, 255, 0, cv2
.NORM_MINMAX , cv2.CV_8UC1)
    bar4.next()

    if auxVers == 'DNG':
        imatge_norm = cv2.cvtColor(imatge_norm, cv2.COLOR_RGB2
BGR)
    bar4.next()

    cv2.imwrite("pinhole_" + data_diaria() + ".JPG", imatge_no
rm)
    bar4.next()

    print() #Per que no es juntin els bars

    return None
```