CSE4019 - Image Processing

Digital Assignment

SANJIT C K S
18BCE0715
SLOT - C2

Image processing concept implementation

Question/Task

Implement a program (any programming language) for any of the image processing techniques learnt and derive the results. Upload the sample image considered as an input, the program written and the results (output) obtained.

Overview

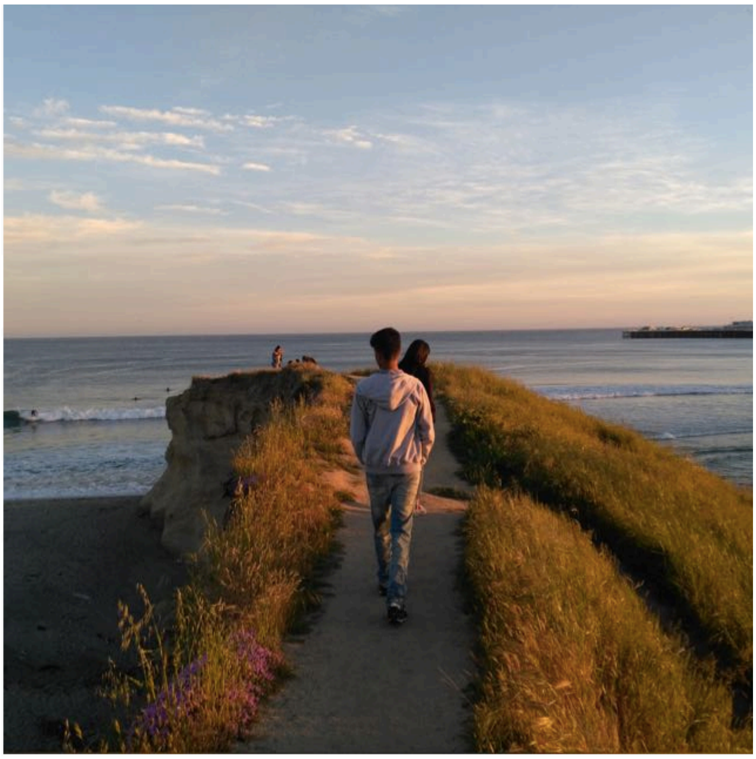
In this Assignment I have chosen a few Spacial Operations to implement. The following Intensity Transformation functions have been implemented with Python 3.7.7 without Open CV. All formulae and methodology were obtained from class notes.

1. Image Negative
2. Log Transform
3. Power-Law Transform
4. Contrast Stretching
5. Graylevel Slicing
6. Bit Plane Splicing

The program takes in input as the operation to perform and the required parameters for that corresponding transformation. It also takes the input image's path as command line argument. Detailed description on how to run has been documented in README.md file of GitHub Repository - https://github.com/sanjitk7/imageManipulationPython

Please Give the above GitHub Repository a look, it has all the sample input and outputs with detailed walkthrough.

Input Sample (Converted to GrayScale on Input)



Source Code/ Implementation

Main.py

```
from PIL import Image
from math import log10
import sys
import argparse
import numpy as np
from image_negetive import image_negetive
from image_log_transform import image_log_transform
from image_gamma_powerlaw_transform import
image_power_law_transform
from contrast_stretching import contrast_stretching
from image_graylevel_slicing import image_graylevel_slicing
from image_bitplane_slicing import bit_plane_splicing
```

```
parser = argparse.ArgumentParser()
parser.add_argument("--input","--picture",help="input image")
args = parser.parse_args()

im = Image.open(args.input).convert("LA")


print("Enter your preferred Operation:\n1.Image Negative\n2.Log Transform\n3.Power-Law Transform\n4.Contrast Stretching\n5.Graylevel Slicing\n6.Bit Plane Splicing")
selection = int(input())

if (selection == 1):
    # SPACIAL IMAGE OPERATIONS - SINGLE PIXEL OPERATIONS
    # 1. IMG NEGETIVE
    print("Formula: s=L-1-r")
    image_negetive(im)
elif (selection == 2):
    # 2. LOG TRANSFORMS
    c = int(input("Enter c value:"))
    print("Formula: s=c*log(1+r)")
    image_log_transform(im,c)
elif(selection==3):
    # 3. POWER-LAW TRANSFORMATIONS
    c = int(input("Enter c value:"))
    gamma = int(input("Enter Gamma value:"))
    print("Formula: s=c*r**gamma")
    image_power_law_transform(im,c,gamma)
elif (selection==4):
    # PIECE WISE OPERATIONS
    # 1. CONTRAST STRETCHING
    r1,s1,r2,s2 = map(int,input("Enter r1, s1, r2, s2 values (with spaces):").split())
    print("Note for binary image conversion use r1=r2")
    print("Formula: s=(r-r1)*((s2-s1)/(r2-r1))+r1")
    contrast_stretching(im,r1,s1,r2,s2)
elif (selection==5):
    # 2. GRAYLEVEL SLICING
    A, B = map(int, input("Enter lower and upper limits:").split())
    S = int(input("Enter S value (conv):"))
    if (input("Enter 'y' if background substitution is required:")=='y'):
        with_bg_subsititution = True
        some_val_bg = 0
    else:
        with_bg_subsititution = False
        some_val_bg = int(input("Enter value to clip the background to:"))

image_graylevel_slicing(im,with_bg_subsititution,some_val_bg,A,B,S)
elif (selection==6):
    # 3. BITPLANE SLICING
    bitPlaneNumber = int(input("Enter the bitplane number to obtain"))
    bit_plane_splicing(im,bitPlaneNumber)
```

Other Utility Functions (Transformation Functions)

image_negetive.py

```
# 1. IMG NEGETIVE
def image_negetive(im):
    L = 256
    pixelMapIn = im.load()
    out1 = im.copy()
    pixelMapOut = out1.load()
    for i in range (im.size[0]):
        for j in range(im.size[1]):
            r = pixelMapIn[i,j][0]
            pixelMapOut[i,j] = (int(L-1-r),pixelMapIn[i,j][1])

    out1.show("Image Negetive")
```

image_log_tranform.py

```
import numpy as np
```

```
# 2. LOG TRANSFORMS
def image_log_transform(im,c):
    pixelMapIn = im.load()
    out1 = im.copy()
    pixelMapOut = out1.load()
    for i in range (im.size[0]):
        for j in range(im.size[1]):
            r = pixelMapIn[i,j][0]
            pixelMapOut[i,j] = (int(c*np.log(1 + 
r)),pixelMapIn[i,j][1])
            # print(pixelMap[i,j])
    out1.show("Log Transform")
```

image_gamma_powerlaw_transform.py

```
# 3. POWER-LAW TRANSFORMATIONS
def image_power_law_transform(im,c=1,gamma=4):
    pixelMapIn = im.load()
    out1 = im.copy()
    pixelMapOut = out1.load()
    for i in range (im.size[0]):
        for j in range(im.size[1]):
            r = pixelMapIn[i,j][0]
            pixelMapOut[i,j] = (int(c*255*((r/255)**(1/
gamma))),pixelMapIn[i,j][1])
    out1.show()
```

contrast_stretching.py

```
def contrast_stretching(im,r1,s1,r2,s2):
    pixelMapIn = im.load()
    out1 = im.copy()
    pixelMapOut = out1.load()
    for i in range(im.size[0]):
        for j in range (im.size[1]):
            r = pixelMapIn[i,j][0]
            if (r1!=r2):
                pixelMapOut[i,j] = (int((r-r1)*(s2-s1)/(r2-r1))+s1,pixelMapIn[i,j][1])
            else:
                pixelMapOut[i,j] = (int((r-r1)*(s2-s1))+s1,pixelMapIn[i,j][1])
    out1.show("Contrast Stretching")
```

image_graylevel_slicing.py

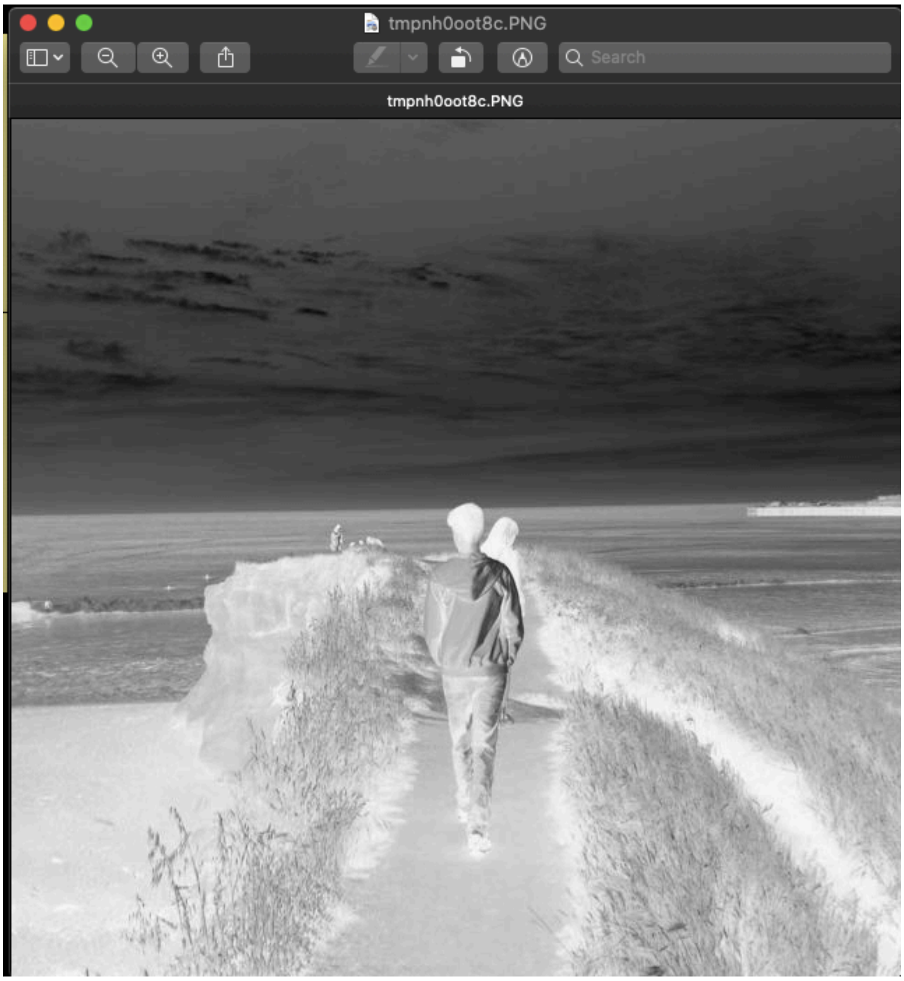
```
# 2. GRAYLEVEL SLICING

def image_graylevel_slicing(im,with_bg_subsititution,A,B,S,some_val_bg=0):
    pixelMapIn = im.load()
    out1 = im.copy()
    pixelMapOut = out1.load()
    for i in range (im.size[0]):
        for j in range(im.size[1]):
            r = pixelMapIn[i,j][0]
            if (r>=A and r<=B):
                pixelMapOut[i,j] = (S,pixelMapIn[i,j][1])
            elif(with_bg_subsititution):
                pixelMapOut[i,j] = (r,pixelMapIn[i,j][1])
            else:
                pixelMapOut[i,j] = (some_val_bg,pixelMapIn[i,j][1])
    out1.show()
```

image_bitplane_slicing.py

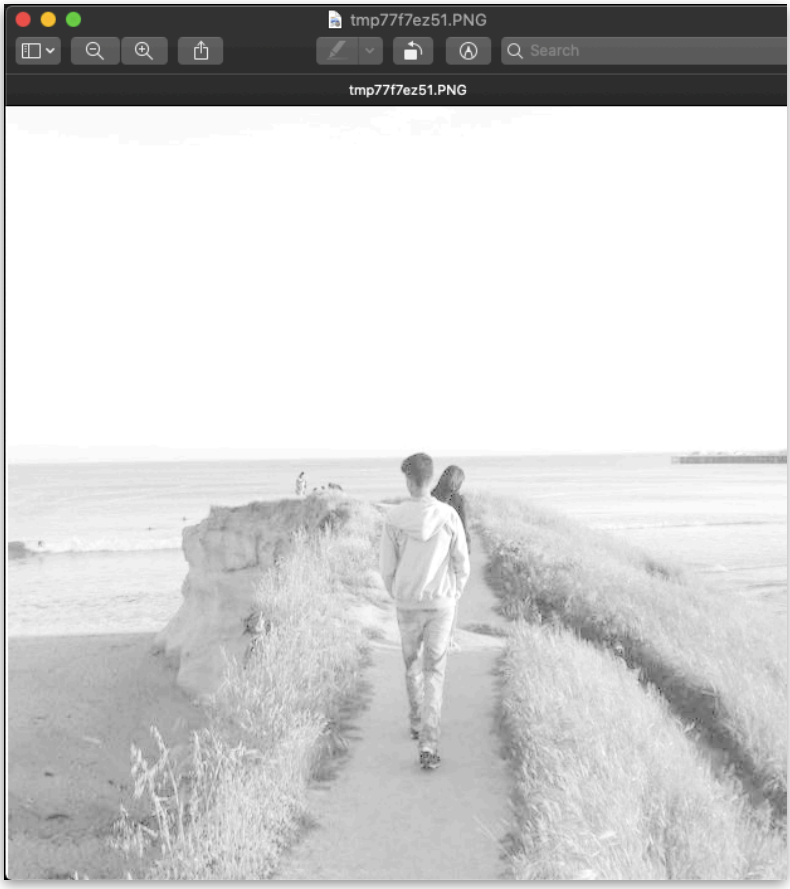
```
import numpy as np

# BIT PLANE SPLICING

def bit_plane_splicing(im,bitPlaneNumber):
    pixelMapIn = im.load()
    out1 = im.copy()
    pixelMapOut = out1.load()
    bitPlaneIndex = 8-bitPlaneNumber
    multiplication_factor = 2**(bitPlaneNumber-1)
    for i in range(im.size[0]):
        for j in range (im.size[1]):
            r = pixelMapIn[i,j][0]
            bin_r = np.binary_repr(r,width=8)
            # print("r,bin_r:",r,bin_r[bitPlaneNumber])
            pixelMapOut[i,j] = 
(int(bin_r[bitPlaneIndex])*multiplication_factor,pixelMapIn[i,j]
[1])
    out1.show()
```

Output of the Above Transformations

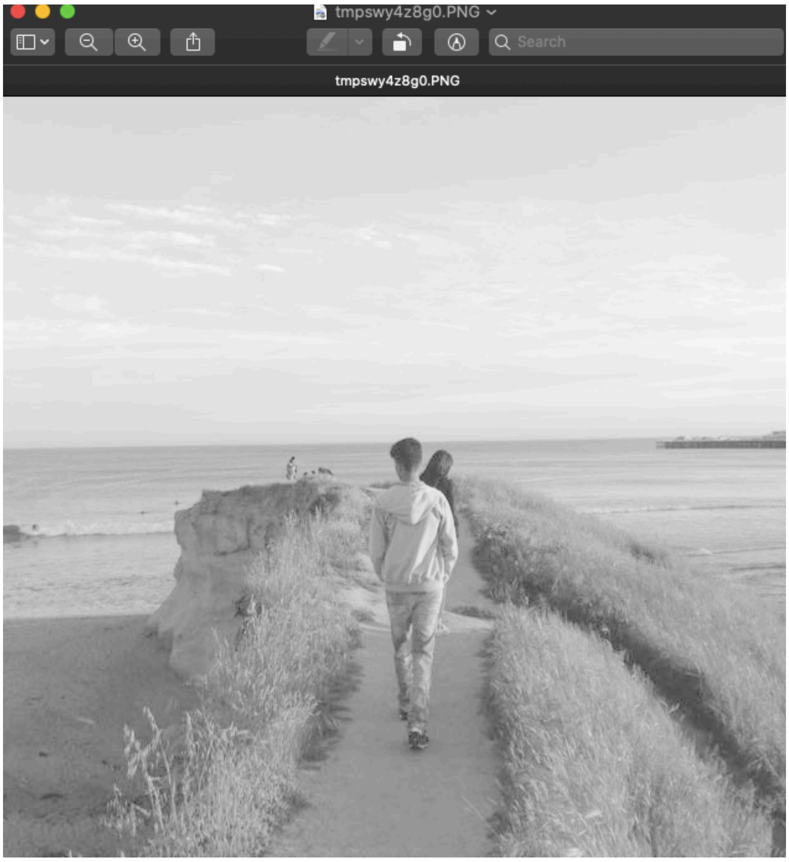
1. Image Negative



2. Log Transform with c=50

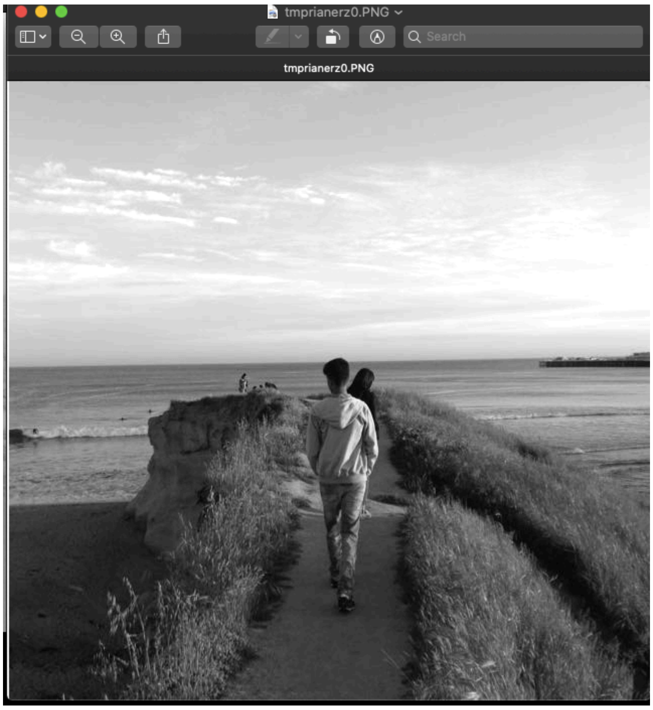


3. Power-Law Transform with c=1 and $\gamma = 3$

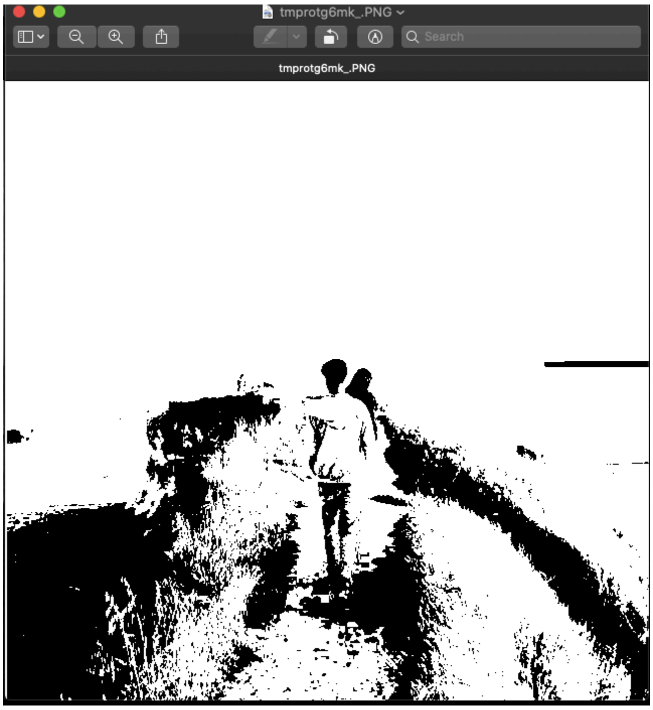


4. Contrast Stretching

with r1=50, s1=60, r2=150, s2=180

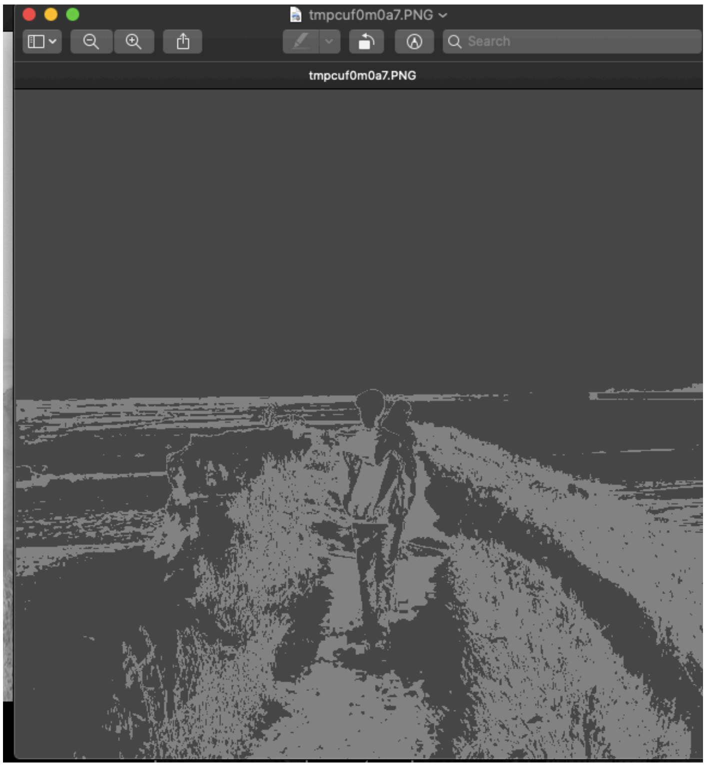


with r1=50, s1=1, r2=50, s2=255
(Thresholding i.e it gives Binary Image)

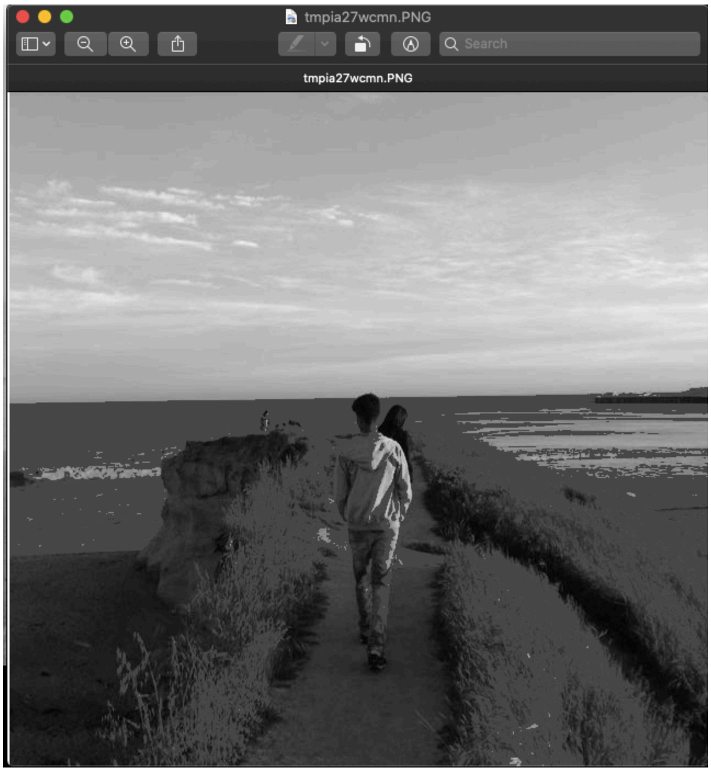


5. Graylevel Slicing

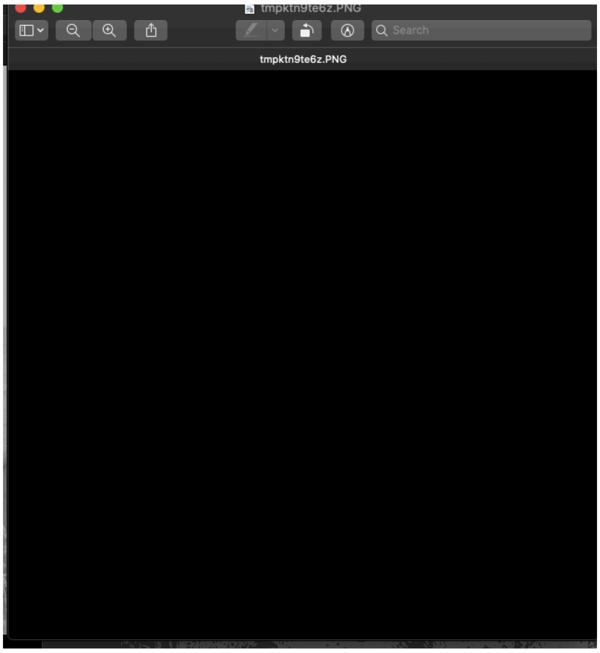
a. without Background Slicing and limits = 80 to 130 and S=70 and background-clipped to 50

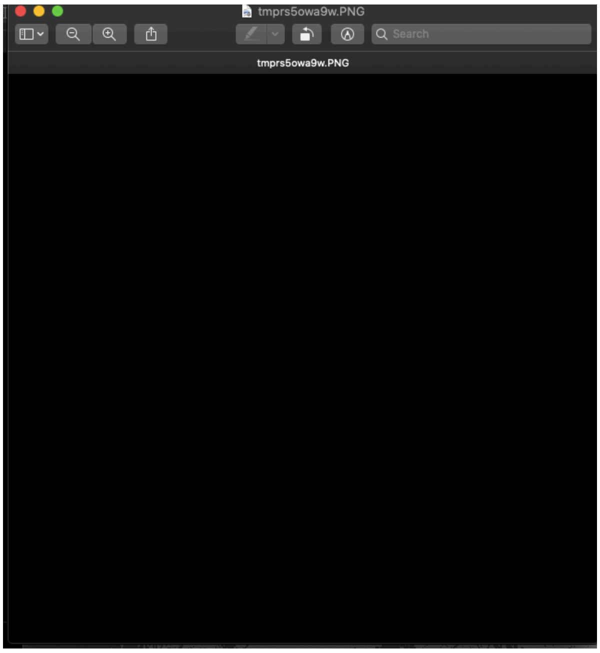


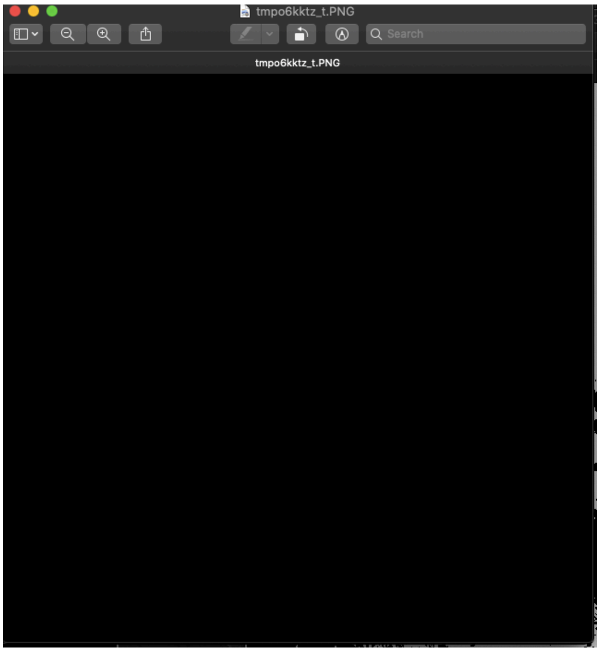
b. with Background Slicing and with limits = 80 to 130 and S=70

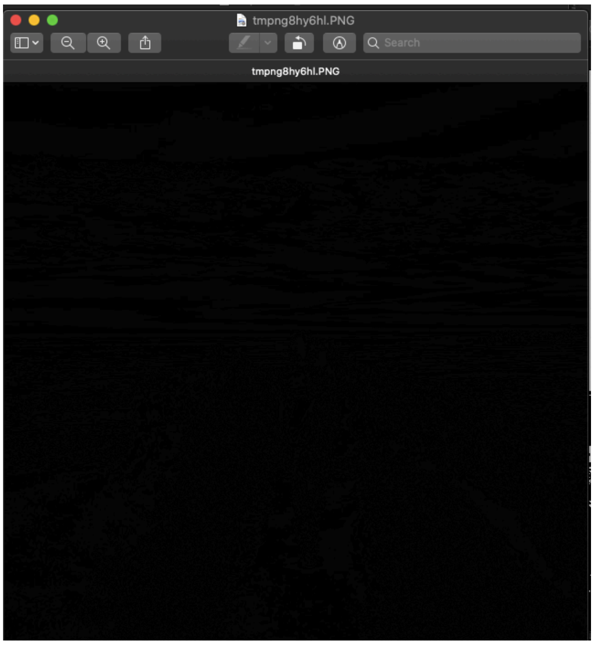


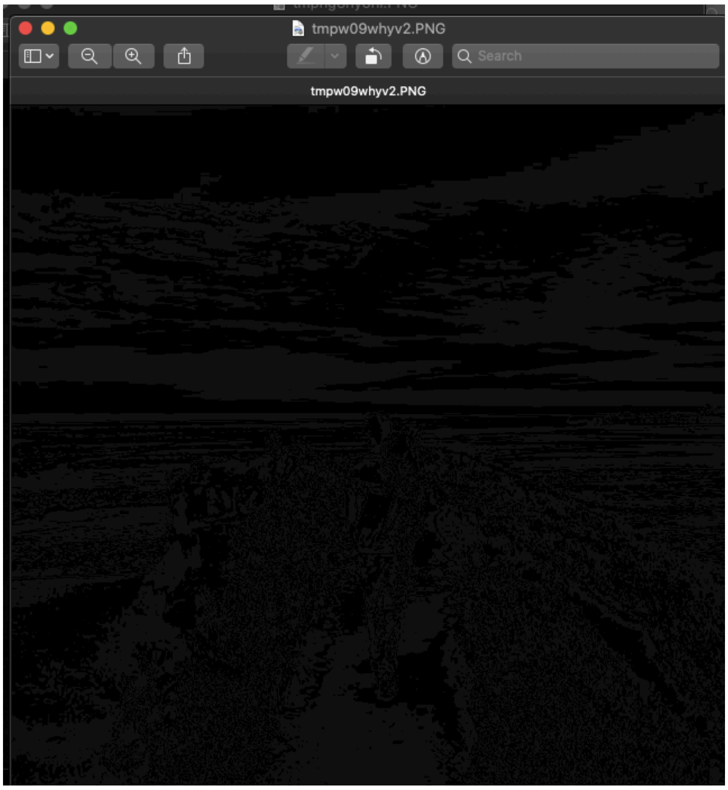
6. Bit Plane Slicing (from plane 1 to 8 - LSB to MSB - Left to Right and Top to Bottom)

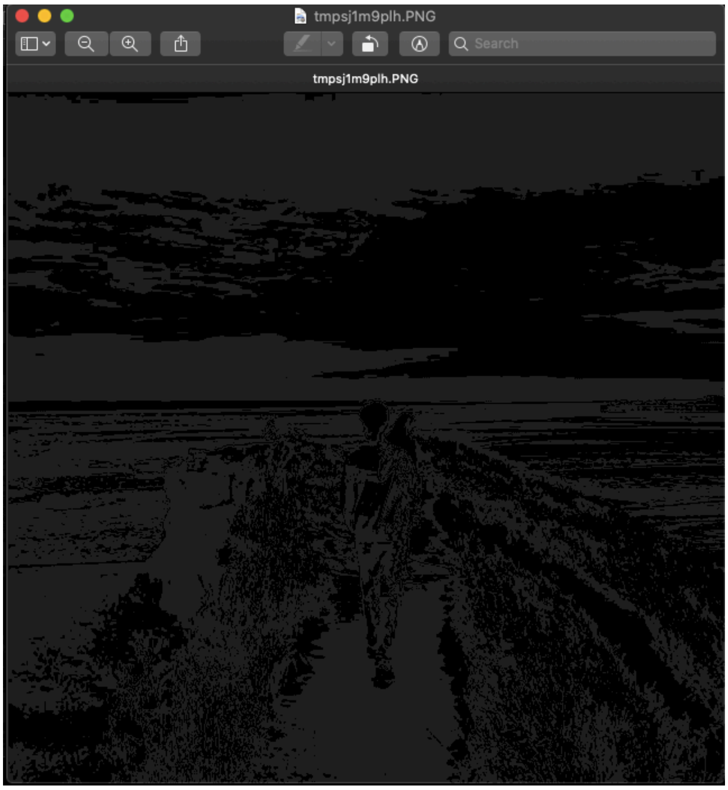


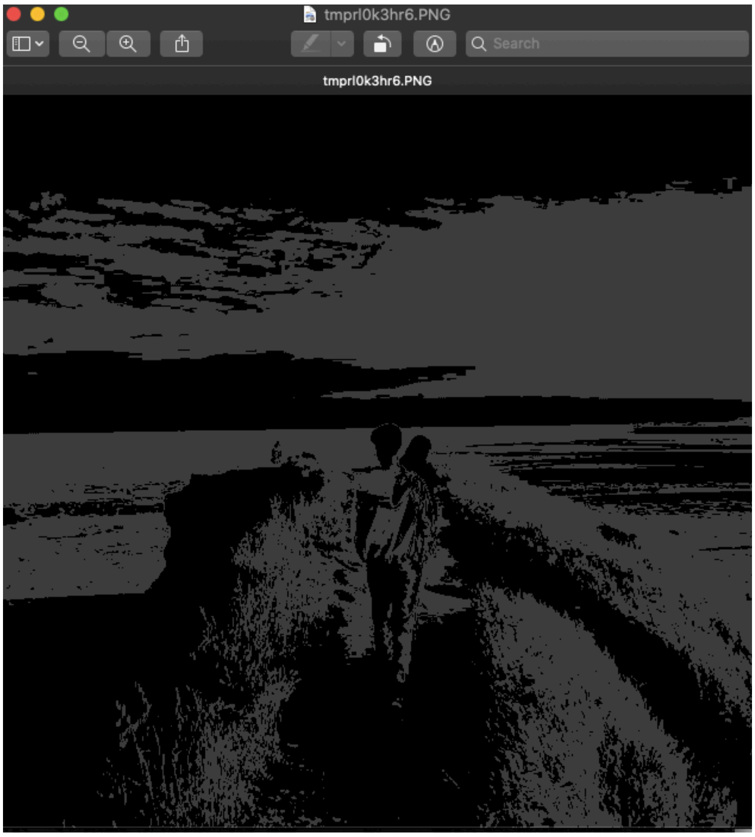


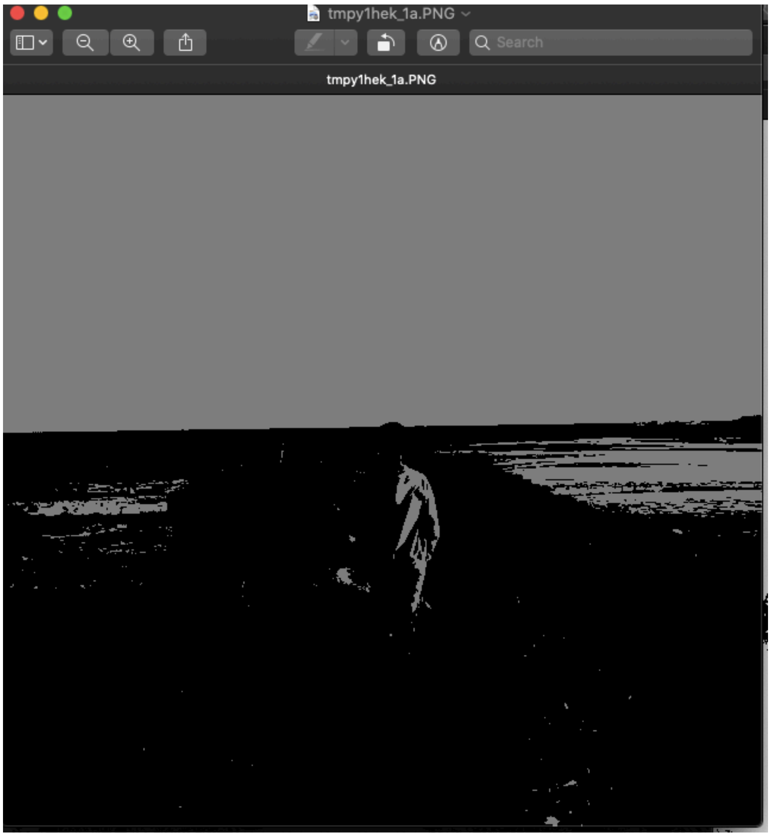





tmpw09whyv2.PNG
Search
tmpw09whyv2.PNG


tmpsj1m9plh.PNG
Search
tmpsj1m9plh.PNG


tmprl0k3hr6.PNG
Search
tmprl0k3hr6.PNG


tmpy1hek_1a.PNG
Search
tmpy1hek_1a.PNG