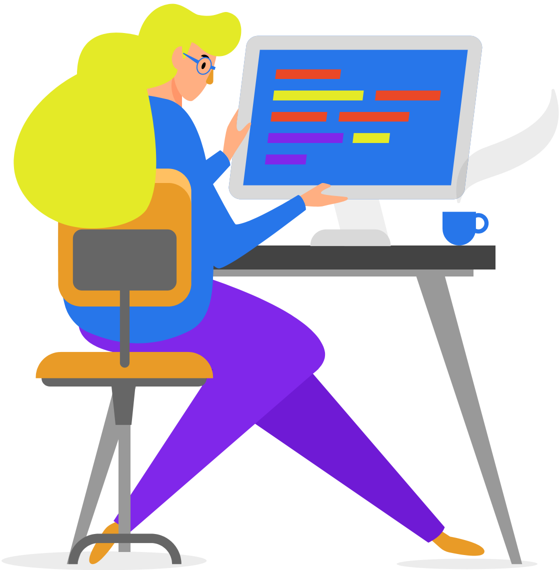
Special Effects in Digital Image Processing

By:
Sumaita Binte Shorif

OUTLINE



01 **Intro to Special Effects**

Definition, Purpose

02 **Radial Pixelation**

03 **Ripple Effects**

Bathroom Glass Ripple, Pond Ripple

04 **General Distortion Effects**

Fisheye, Twirl, Jitter, Circular Slice, Square Slice

05 **Pixel Effects**

Oil Painting, Solarization

06 **Effects on Color Images**

Twirl Effect, Ripple Effect

Introduction

Definition

- A process of changing either of the value or the position of pixels in an image.

Purpose of Applying Special Effects

- To make the image more better to analyze
- To make the image look better
- Just for fun
- To make the image more dramatic

Radial Pixelation

- An image can be “pixelated”; that is, be shown in large blocks of low resolution, by use of the “imresize” function.

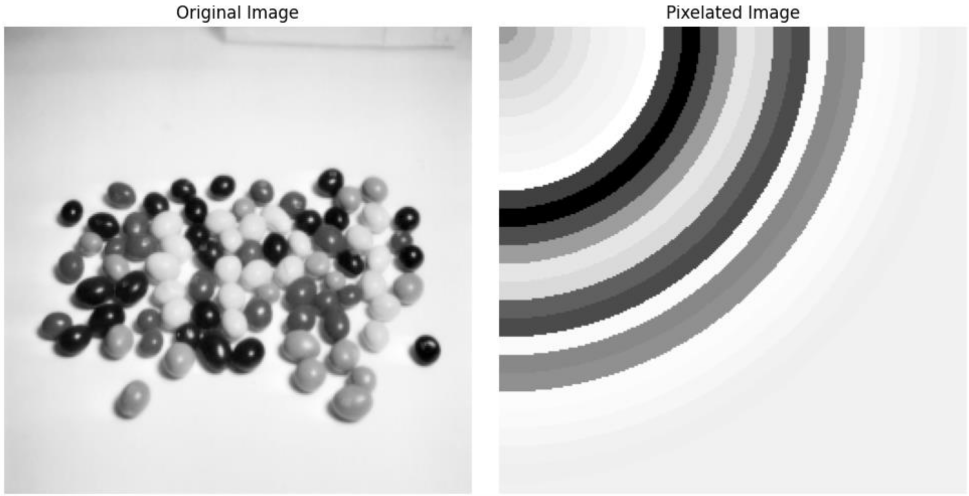
- We can achieve the same effect, however, by using the mod function.

- In general, mod(x, n) is the remainder when x is divided by n.

Radial Pixelation (Cont.)

```
import numpy as np
from skimage import io, color
import matplotlib.pyplot as plt

f = io.imread('C:\\Dataset\\4.1.08.tiff')
fg = color.rgb2gray(f)
rows, cols = fg.shape

ox = oy = 0
y, x = np.indices((rows, cols))
r = np.sqrt((x - ox)**2 + (y - oy)**2)

step = 10
x2 = np.clip(((r // step) * step + step // 2), 0, cols - 1).astype(int)
y2 = x2

f2 = fg[y2, x2]

plt.figure(figsize=(10, 5))

plt.subplot(121, title='Original Image')
plt.imshow(fg, cmap='gray'), plt.axis('off')

plt.subplot(122, title='Pixelated Image')
plt.imshow(f2, cmap='gray'), plt.axis('off')

plt.tight_layout()
plt.show()
```

Python Code



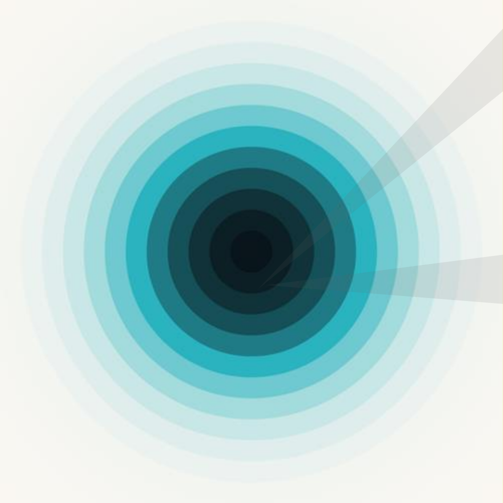
Original Image

Pixelated Image

Output Obtained

Ripple Effect

We will Investigate 2 Different Ripple Effects:



Bathroom Glass

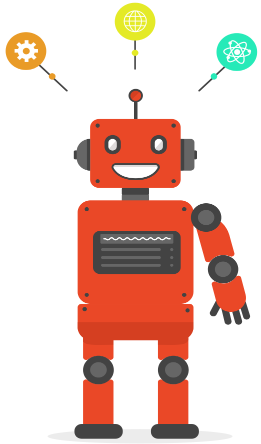
Ripples, which give the effect of an image seen through wavy glass, such as is found in bathrooms

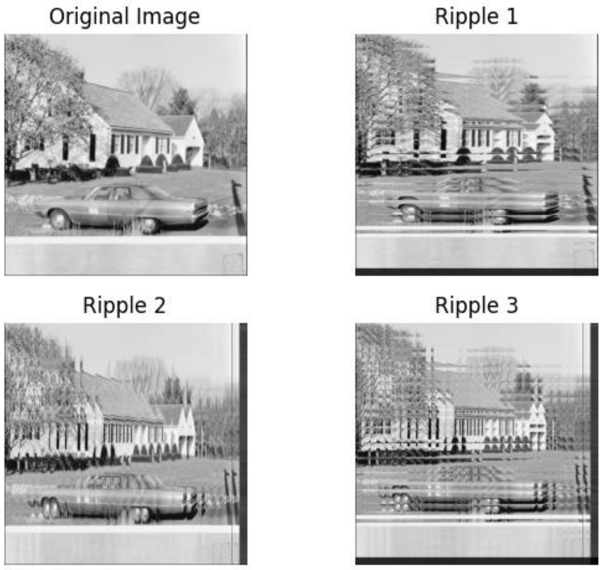
Pond Ripple

Which approximates a reflection on the surface of a pond.

Bathroom Glass Ripple Effect

```
import numpy as np
from skimage import io, color
import matplotlib.pyplot as plt

f = io.imread('C://Dataset//house.tiff')
fg = color.rgb2gray(f)
rows, cols = fg.shape

def clip(x, y, z):
    return np.where(np.where(x < y, y, x) > z, z, x)

y, x = np.indices((cols, rows))

x2 = clip(x + x % 32, 0, rows - 1)
ripple1 = np.reshape(fg[x2.ravel(), y.ravel()], (cols, rows)).T

y2 = clip(y + y % 32, 0, cols - 1)
ripple2 = np.reshape(fg[x.ravel(), y2.ravel()], (rows, cols)).T

y3 = clip(y + y % 32, 0, cols - 1)
ripple3 = np.reshape(fg[x2.ravel(), y3.ravel()], (cols, rows)).T

plt.figure(figsize=(10, 8))
plt.subplot(2, 2, 1), plt.imshow(fg, cmap='gray'), plt.title('Original
Image'), plt.axis('off')
plt.subplot(2, 2, 2), plt.imshow(ripple1, cmap='gray'), plt.title('Ripple 1'),
plt.axis('off')
plt.subplot(2, 2, 3), plt.imshow(ripple2, cmap='gray'), plt.title('Ripple 2'),
plt.axis('off')
plt.subplot(2, 2, 4), plt.imshow(ripple3, cmap='gray'), plt.title('Ripple 3'),
plt.axis('off')
plt.tight_layout()
plt.show()
```



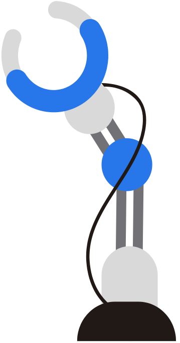


Python Code

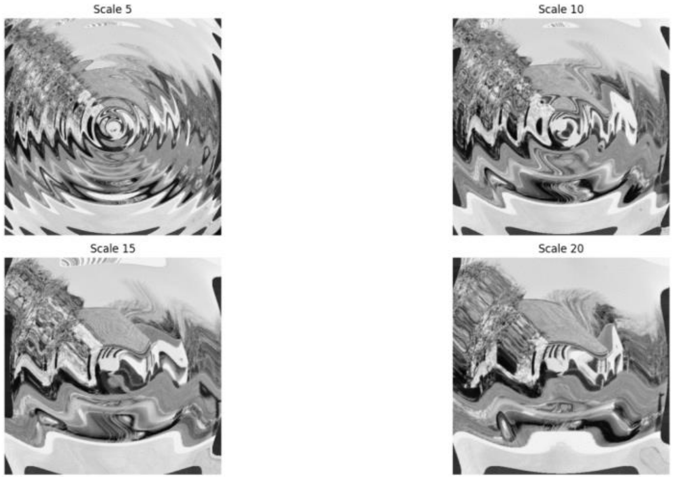
Output Obtained

Pond Ripple Effect

```
import numpy as np
import matplotlib.pyplot as plt

image = plt.imread('C://Dataset//house.tiff')
gray_image = np.mean(image, axis=2)

x, y = np.mgrid[0:gray_image.shape[0], 0:gray_image.shape[1]]

center_x, center_y = gray_image.shape[1] // 2, gray_image.shape[0] //
2

scale_values = [5, 10, 15, 20]
shift_range = 20

plt.figure(figsize=(10, 8))

for i, scale in enumerate(scale_values, start=1):
    distance = np.sqrt((x - center_x)**2 + (y - center_y)**2)
    shifts = (np.sin(distance / scale) * shift_range).astype(int)
    x_shifted = np.clip(x + shifts, 0, gray_image.shape[0] - 1)
    y_shifted = np.clip(y + shifts, 0, gray_image.shape[1] - 1)
    ripple_effect = gray_image[x_shifted, y_shifted]

    plt.subplot(2, 2, i), plt.imshow(ripple_effect, cmap='gray'),
plt.axis('off')
    plt.title(f'Scale {scale}')

plt.tight_layout()
plt.show()
```



Python Code



Scale 5

Scale 10

Scale 15

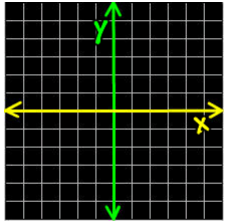
Scale 20

Output Obtained

General Distortion of Effects

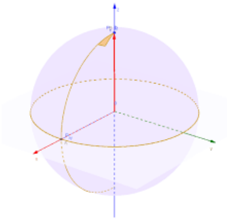
We can calculate general distortion effect in 2 ways:

01



Cartesian Coordinates

02



Polar Coordinates

Types of General Distortion Effects

01 Fisheye

Twirl 02

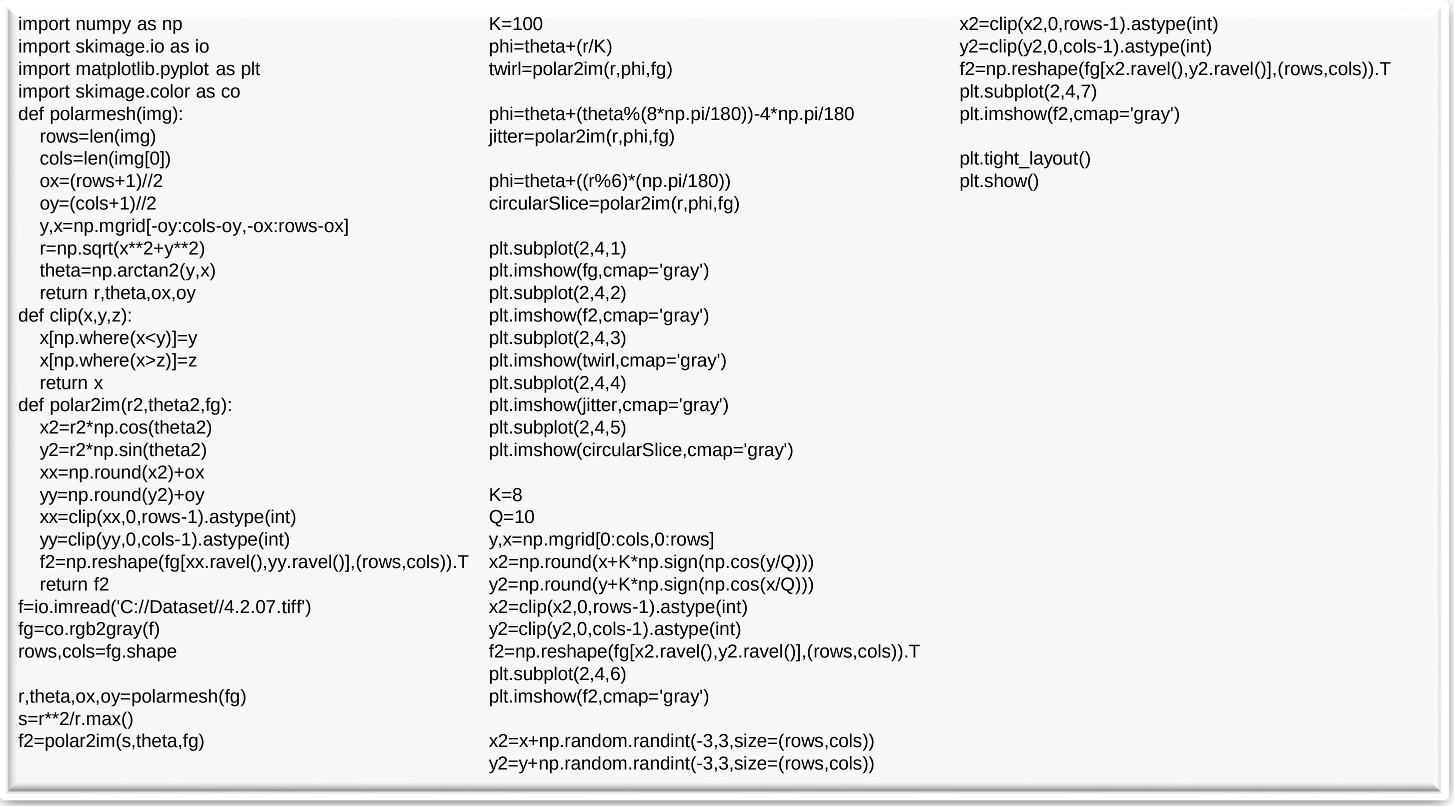
03 Jitter

Circular Slice 04

05 Square Slice

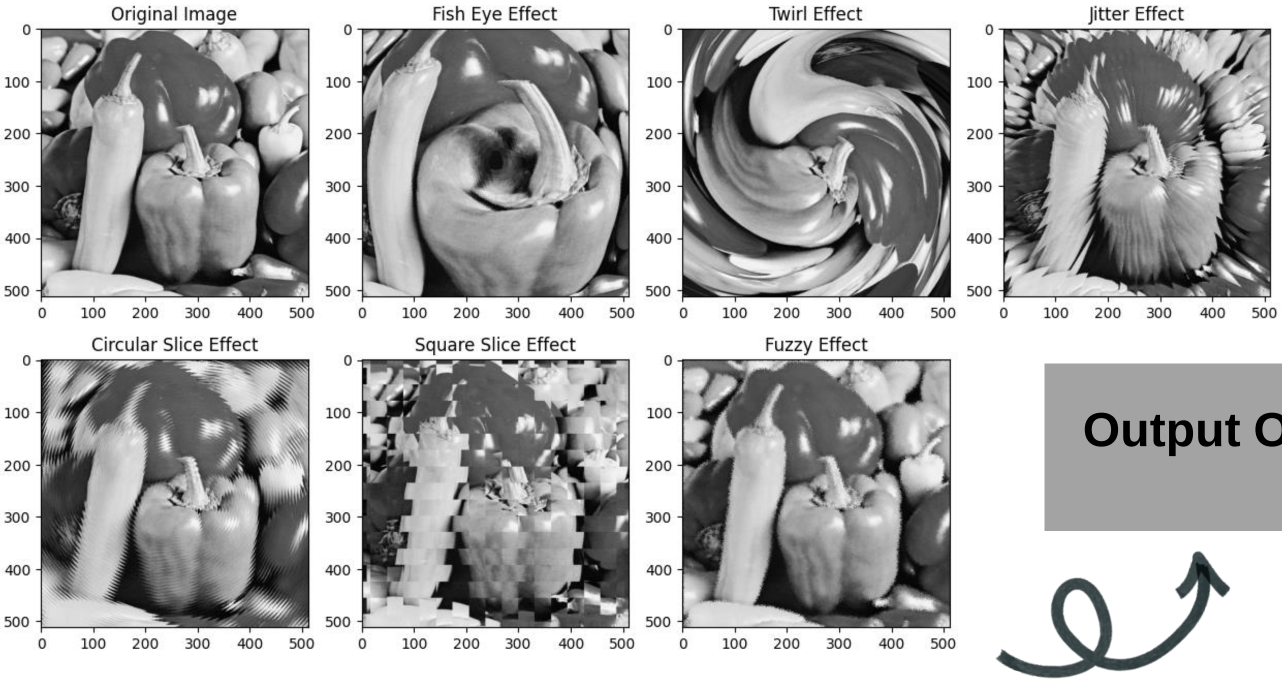
Fuzzy Effect 06

Python Code for General Distortion Effects



```
import numpy as np
import skimage.io as io
import matplotlib.pyplot as plt
import skimage.color as co
def polarmesh(img):
    rows=len(img)
    cols=len(img[0])
    ox=(rows+1)//2
    oy=(cols+1)//2
    y,x=np.mgrid[-oy:cols-oy,-ox:rows-ox]
    r=np.sqrt(x**2+y**2)
    theta=np.arctan2(y,x)
    return r,theta,ox,oy
def clip(x,y,z):
    x[np.where(x<y)]=y
    x[np.where(x>z)]=z
    return x
def polar2im(r2,theta2,fg):
    x2=r2*np.cos(theta2)
    y2=r2*np.sin(theta2)
    xx=np.round(x2)+ox
    yy=np.round(y2)+oy
    xx=clip(xx,0,rows-1).astype(int)
    yy=clip(yy,0,cols-1).astype(int)
    f2=np.reshape(fg[xx.ravel(),yy.ravel()],(rows,cols)).T
    return f2
f=io.imread('C://Dataset//4.2.07.tiff')
fg=co.rgb2gray(f)
rows,cols=fg.shape

r,theta,ox,oy=polarmesh(fg)
s=r**2/r.max()
f2=polar2im(s,theta,fg)
K=100
phi=theta+(r/K)
twirl=polar2im(r,phi,fg)

phi=theta+(theta%(8*np.pi/180))-4*np.pi/180
jitter=polar2im(r,phi,fg)

phi=theta+((r%6)*(np.pi/180))
circularSlice=polar2im(r,phi,fg)

plt.subplot(2,4,1)
plt.imshow(fg,cmap='gray')
plt.subplot(2,4,2)
plt.imshow(f2,cmap='gray')
plt.subplot(2,4,3)
plt.imshow(twirl,cmap='gray')
plt.subplot(2,4,4)
plt.imshow(jitter,cmap='gray')
plt.subplot(2,4,5)
plt.imshow(circularSlice,cmap='gray')

K=8
Q=10
y,x=np.mgrid[0:cols,0:rows]
x2=np.round(x+K*np.sign(np.cos(y/Q)))
y2=np.round(y+K*np.sign(np.cos(x/Q)))
x2=clip(x2,0,rows-1).astype(int)
y2=clip(y2,0,cols-1).astype(int)
f2=np.reshape(fg[x2.ravel(),y2.ravel()],(rows,cols)).T
plt.subplot(2,4,6)
plt.imshow(f2,cmap='gray')

x2=x+np.random.randint(-3,3,size=(rows,cols))
y2=y+np.random.randint(-3,3,size=(rows,cols))
x2=clip(x2,0,rows-1).astype(int)
y2=clip(y2,0,cols-1).astype(int)
f2=np.reshape(fg[x2.ravel(),y2.ravel()],(rows,cols)).T
plt.subplot(2,4,7)
plt.imshow(f2,cmap='gray')

plt.tight_layout()
plt.show()
```

General Distortion Effects (Cont.)



Original Image

Fish Eye Effect

Twirl Effect

Jitter Effect

Circular Slice Effect

Square Slice Effect

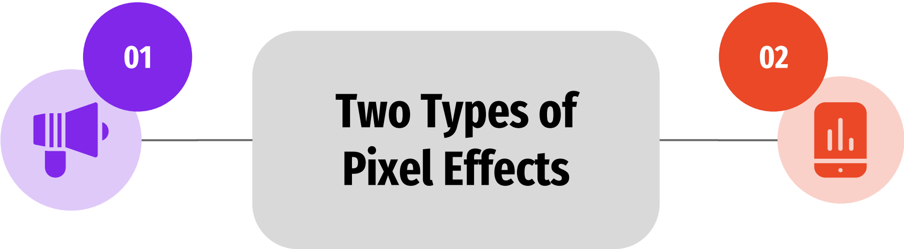
Fuzzy Effect

Output Obtained

Pixel Effects

- Pixel Effects In a sense, all image effects are “pixel effects,” because we are dealing with pixels.

- Many effects take the pixel values and apply some sort of processing routines to them.



01

Two Types of Pixel Effects

02

Oil Painting

it works by means of a non-linear filter; the output of the filter is the most common pixel value in the filter mask.

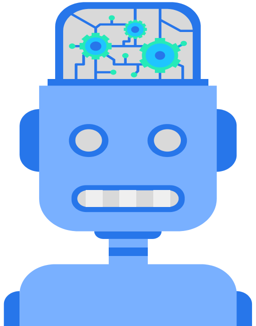
Solarization

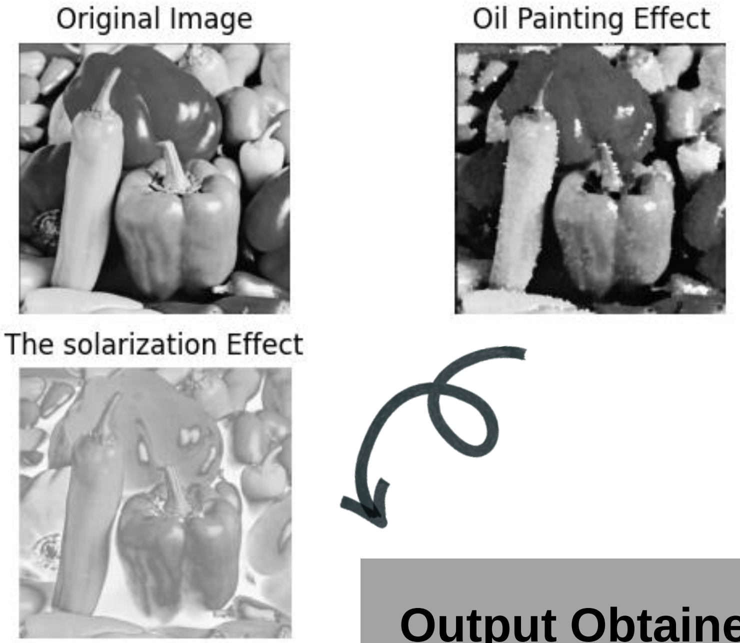
This is a photographic effect that is obtained by applying diffuse light to a developing photograph, and then continuing with the development.

Pixel Effects (Cont.)

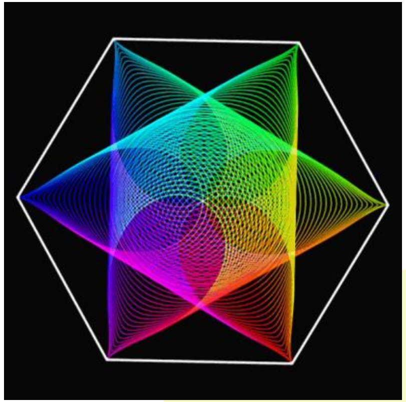
```
import numpy as np
import skimage.io as io
import matplotlib.pyplot as plt
import skimage.color as co
import scipy.stats as stats
import scipy.ndimage as ndi

def mymode(x):
    return stats.mode(x, axis=None, keepdims=True)[0][0]
f = io.imread('C://Dataset//4.2.07.tiff')
fg = co.rgb2gray(f)
f2 = ndi.generic_filter(fg, mymode, size=(9, 9))
u=fg>0.5
sol=u*fg+(1-u)*(1-fg)
plt.subplot(2, 2, 1)
plt.imshow(fg, cmap='gray')
plt.axis('off')
plt.title('Original Image')
plt.subplot(2, 2, 2)
plt.axis('off')
plt.imshow(f2, cmap='gray')
plt.title('Oil Painting Effect')
plt.subplot(2, 2, 3)
plt.axis('off')
plt.imshow(sol,vmin=0,vmax=1, cmap='gray')
plt.title('The solarization Effect')
plt.show()
```

Python Code





Output Obtained

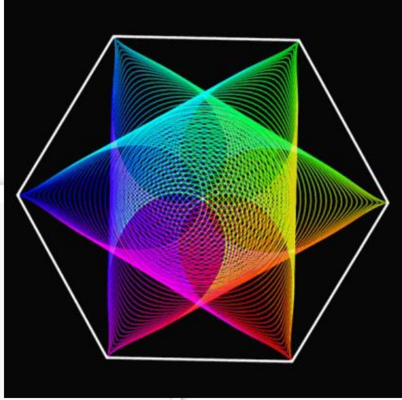


Color Images

Application to a color image means applying the effect to each of the RGB components separately
In order to maintain the colors, we need to shift all the RGB components.

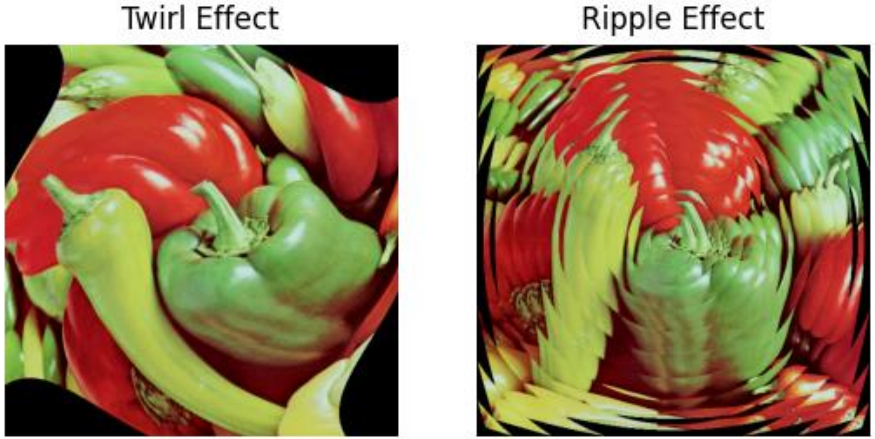
Here we will see twirl and circular ripple applied to color images.

Python Code Color Images



```
import numpy as np
import skimage.io as io
import matplotlib.pyplot as plt
import skimage.color as co
import scipy.ndimage as ndi

def polarmesh(img):
    rows = len(img)
    cols = len(img[0])
    ox = (rows + 1) // 2
    oy = (cols + 1) // 2
    y, x = np.mgrid[-oy:cols - oy, -ox:rows - ox]
    r = np.sqrt(x**2 + y**2)
    theta = np.arctan2(y, x)
    return r, theta, ox, oy

def polar2im(img, r, phi):
    rows, cols = img.shape
    ox = cols // 2
    oy = rows // 2
    x = ox + r * np.cos(phi)
    y = oy + r * np.sin(phi)
    return ndi.map_coordinates(img, [y, x], order=3,
mode='constant', cval=0)

f = io.imread('C://Dataset//4.2.07.tiff')
fg=co.rgb2gray(f)

rows,cols=fg.shape

r,theta,ox,oy = polarmesh(fg)

k = 250
phi = theta + r / k
r2 = r + (r % 30)

twirl = np.zeros_like(f)
ripple = np.zeros_like(f)

for i in range(3):
    twirl[:, :, i] = polar2im(f[:, :, i], r, phi)
    ripple[:, :, i] = polar2im(f[:, :, i], r2, theta)

plt.subplot(1, 2, 1)
plt.imshow(twirl)
plt.axis('off')
plt.title('Twirl Effect')
plt.subplot(1, 2, 2)
plt.imshow(ripple)
plt.axis('off')
plt.title('Ripple Effect')
plt.show()
```

Output Obtained for Color Images



Conclusion

There are so many effects to apply on images but we have only covered a few.
Special effects on images makes the image more useful to analyze.
Also, we can apply these effects for fun.
Effects on RGB images are more dramatic than Gray color images.

Thanks!

Questions? Comments? Let us Know