```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import skimage.io as io
import cv2


sig_o = 12.5  # Need this to get similar-looking results


# Build Laplacian Pyramid
def createPyramid( image, sig_o ):

    pyramid = []
    sig = sig_o
    curImage = image

    # Uses N+1 levels of a pyramid
    N = 5
    for _ in range(N):

        prevImage = curImage
        height, width, _ = tf.shape( curImage )

        # Gaussian Pyramid
        # Apply blur and downsample (and apply blur again)
        curImage = cv2.pyrDown( curImage )
        curImage = applyGauss( curImage, sig )

        # Laplacian Pyramid
        # Upsample the blurred image and compute the difference
        resizedImage = cv2.pyrUp( curImage, dstsize=(height, width) )
        pyramid.append( prevImage - resizedImage )

        sig = sig*2

    # Append top of pyramid
    pyramid.append( curImage )

    return pyramid

def applyGauss( image, sig ):

    # Apply Gaussian blur
```

```
    ksize = (0,0)   # Kernel size is computed from sig values
    sigX = sig
    sigY = sigX     # sigY uses sigX
    image = cv2.GaussianBlur( image, ksize, sigX, sigY )

    return image


# Apply weighting to levels of a pyramid
def expWeight( laplacePyramid, u_o, v_o, f_o ):

    # Ensure center is an int
    u_o = tf.cast( np.floor(u_o), tf.int32 )
    v_o = tf.cast( np.floor(v_o), tf.int32 )
    f_o = tf.cast( np.floor(f_o), tf.int32 )

    # Get ratios to use for each level
    height, width, _ = tf.shape( laplacePyramid[0] )
    u_o = tf.cast( u_o/height, tf.float32 )
    v_o = tf.cast( v_o/width, tf.float32 )

    pyramid = []

    for k in range( len(laplacePyramid) ):

        height, width, _ = tf.shape( laplacePyramid[k] )
        height = tf.cast( height, tf.float32 )
        width = tf.cast( width, tf.float32 )

        # Ensure center is an int
        u_cur = np.ceil(u_o * height)
        v_cur = np.ceil(v_o * width)

        # Generate coordinates centered at (u_o, v_o) for each level
        u_front = tf.reverse( tf.range( 1, u_cur ), axis=[-1] )
        u_back = tf.range( height-u_cur+1 )
        v_front = tf.reverse( tf.range( 1, v_cur ), axis=[-1] )
        v_back = tf.range( width-v_cur+1 )
        u = tf.concat( [ u_front, u_back ], axis=-1 )
        v = tf.concat( [ v_front, v_back ], axis=-1 )

        # Create a matrix of coordinates
        v = np.expand_dims( v, axis=-1 )            # For broadcasting
        coords = tf.math.pow(u,2) + tf.math.pow(v,2)
```

```
        # Create the foveal size for each level
        f_k = tf.math.pow(2,k) * f_o
        f_k = tf.cast( f_k, tf.float32 )

        # Get the kernel needed to multiply the pyramid
        kernel = tf.exp( -coords / ( 2*tf.math.pow(f_k,2) ) )
        kernel = np.expand_dims( kernel, axis=-1 )  # For broadcasting

        pyramid.append( laplacePyramid[k] * kernel )

    return pyramid


def applyBlur( image, u_o, v_o, f_o ):

    # Convert from Tensor to Numpy Array
    image = image.numpy()

    pyramid = createPyramid( image, sig_o )
    pyramid = expWeight( pyramid, u_o, v_o, f_o )

    # Sum each level of pyramid
    finalImg = pyramid[-1]
    for i in reversed( range( len(pyramid)-1 ) ):

        height, width, _ = tf.shape( pyramid[i] )
        finalImg = cv2.pyrUp( finalImg, dstsize=(height, width) )
        finalImg += pyramid[i]

    return finalImg
```