**Computer Vision**

**Assignment 01**

**Sahithi Muppiri**

**Laplacian Blending:**

Input:

Apple, mask, and orange images

A picture containing sitting, indoor, orange, apple

Description automatically generated Shape

Description automatically generated A close up of a lemon

Description automatically generated with low confidence

Procedure:

1. Three input images are chosen, and they are cropped/resized to same sizes.
2. Gaussian pyramid is constructed for the apple, orange, and mask. This is used to shrink the original size of the image by ½, ¼, and so on.
3. Laplacian image is created for apple and orange gaussian pyramids.
4. Two of the Laplacian images formed are blended using the mask with the gaussian pyramid.
5. The image is then reconstructed to show the blended image.

Some output of the code:

1. Unblended image for apple and orange.

A screenshot of a computer screen

Description automatically generated with low confidence

1. Gaussian Pyramids of orange

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Blending of two Laplacian images (apple)

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

1. Output of the images

A screenshot of a computer screen

Description automatically generated with medium confidence

**Output:**

Blended image of orange and apple

**A close up of an orange

Description automatically generated with medium confidence**

**Code:**

# install the libraries

import numpy as np

from scipy import ndimage

import cv2

import imageio

def upscale(image):

  image\_up = np.zeros((2\*image.shape[0], 2\*image.shape[1]))

  image\_up[::2, ::2] = image

  return ndimage.filters.convolve(image\_up,4, mode='constant')

def downscale(image):

  print(np.shape(image), np.shape())

  image\_blur = ndimage.filters.convolve(image , mode='constant')

  return image\_blur[::2, ::2]

#Build gaussian and laplacian pyramids

def pyramids(image):

  # Initialize pyramids

  Gaussian = [image, ]

  Laplacian = []

  # Build the Gaussian pyramid till the maximum possible region

  while image.shape[0] >= 2 and image.shape[1] >= 2:

    image = downscale(image)

    Gaussian.append(image)

  # Build the Laplacian pyramid

  for i in range(len(Gaussian) - 1):

    Laplacian.append(Gaussian[i] - upscale(Gaussian[i + 1]))

  return Gaussian[:-1], Laplacian

# Build Gaussian pyramid and Laplacian pyramids from apple,  orange and mask

def blending(A, B, mask):

  [G\_apple, L\_apple] = pyramids(A)

  [G\_orange ,L\_orange] = pyramids(B)

  # Build a Gaussian pyramid GR from selected region R

  [g\_mask, l\_mask] = pyramids(mask)

  # Collapse the LS pyramid to get the final blended image

  blend = []

  for i in range(len(L\_apple)):

    laplacian\_sum = g\_mask[i]/255\*L\_apple[i] + (1-g\_mask[i]/255)\*L\_orange[i]

    blend.append(laplacian\_sum)

  return blend

# reconstruct the pyramids as well as upsampling and add up with each level

def reconstruct(pyramid):

  revPyramid = pyramid[::-1]

  stack = revPyramid[0]

  for i in range(1, len(revPyramid)):

    stack = upscale(stack) + revPyramid[i]

  return stack

#Colour blending

def color\_blending(a, o, m):

  aR,aG,aB = cv2.split(a)

  oR,oG,oB = cv2.split(o)

  Red = reconstruct(blending(aR, oR, m))

  Green = reconstruct(blending(aG, oG, m))

  Blue = reconstruct(blending(aB, oB, m))

  output = cv2.merge((Red, Green, Blue))

  imageio.imsave("output.png", output)

  img = cv2.imread("output.png")

  cv2.imshow('result',img)

  cv2.waitKey(0)

  cv2.destroyAllWindows()

apple = imageio.imread('apple.jpg')

orange = imageio.imread('orange.jpg')

mask = cv2.imread('mask.jpg', 0)

color\_blending(apple, orange, mask)