Hacettepe University
Computer Engineering Department

BBM415 IMAGE PROCESSING LAB. - 2023 Fall

ASSIGNMENT 2

Giving Cartoon Effect to Colorful Images

Student name:

Hasan Malkoç

Instructor: Aydın Kaya
TA: Burçak Asal

Student Number:

2200356826

Using Image Pyramid for Image Blending

In the image editing tools such as Adobe Photoshop, GIMP, image blending is the one of most fundamental tasks and used for many purposes. For example, you can prepare a poster for an advertisement or a film. The most important thing for image blending methods is to blend images seamlessly. In other words, for a successful image blending method, seams where images or image regions are stitched must be invisible. There are many ways to blend two or more images. One such approach proposed in is Laplacian Pyramid. Accordingly, to this approach, images are first decomposed into their Laplacian pyramids, and then these images are blended in pyramid levels so that they are seamless.

Approach

The program will take an image as input and a masked image region from another or the same image, producing a blended image. The process involves the following steps:

- Build Laplacian pyramids for each image.
- Build a Gaussian pyramid for each region mask.
- Blend each level of the pyramid using the region mask from the same level:

$$L_i^2 = L_i^1 \cdot R_i \; + \; L_i^2 \; \cdot (1 - \; R_i \,)$$

- Collapse the pyramid to obtain the final blended image.

My Implementation:

First, I take two images <u>with same sizes,</u> first image is the source of the mask, and the second image is the image to apply mask on it. I also set a “mask for blending” image where it contains 0’s and 1’s, where 1’s represents the region of the mask as white pixels, and the rest is black pixels. Example:

Image1

Mask for Blending

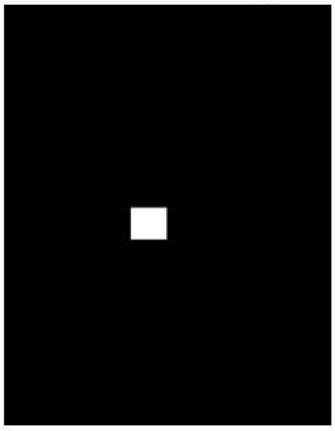


Image2

Shouldn't be confused with labels **Image1** and **Image2**, it is vice versa in the code.

Functions for this implementation:

```
def getMask(mask_image):
def getRegionToMask(input_image):
def setMaskImage(mask_image, region_to_mask, mask):
```

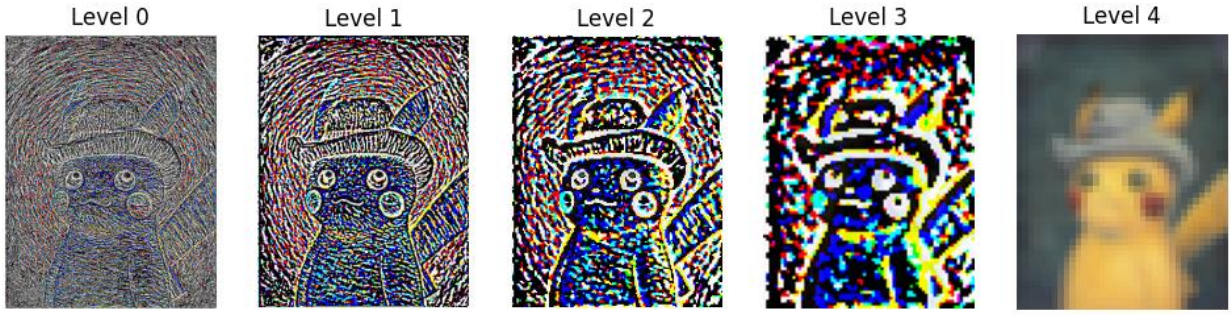
For 4 main steps of the implementation my functions are:

```
def downsample(image):
def upsample(image, target_shape):
def smooth_image(image):
def gaussianPyr(image, number_of_levels):
def laplacianPyr(image, number_of_levels):
def collapsePyr(laplacian_pyramid):
def blend(image1, image2, mask_for_blending, number_of_levels):
```

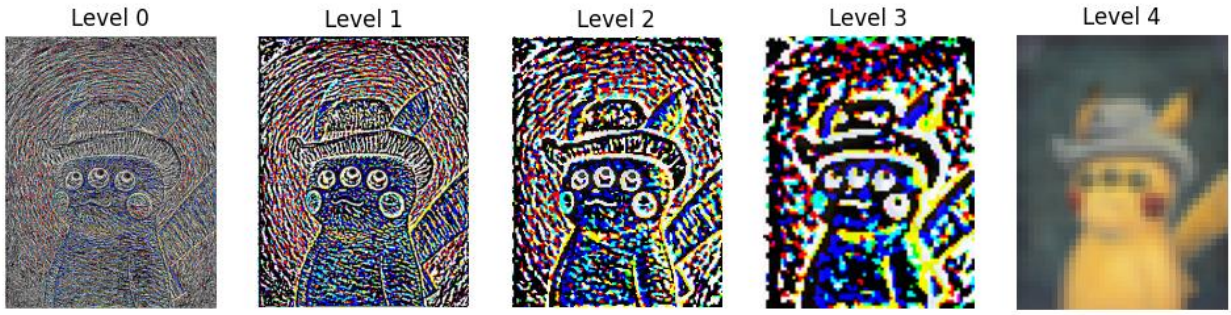
(Detailed implementation is in code/main.py)

Outputs of these 4 main steps are as below: (More examples are after this section)

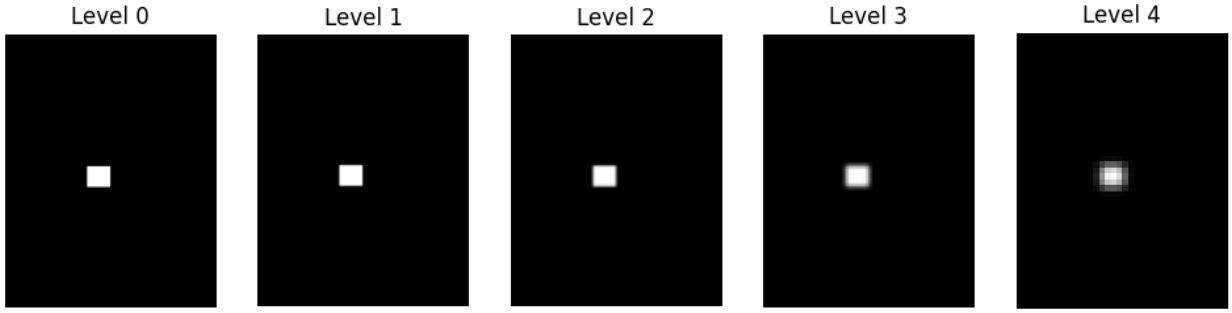
Laplacian 1



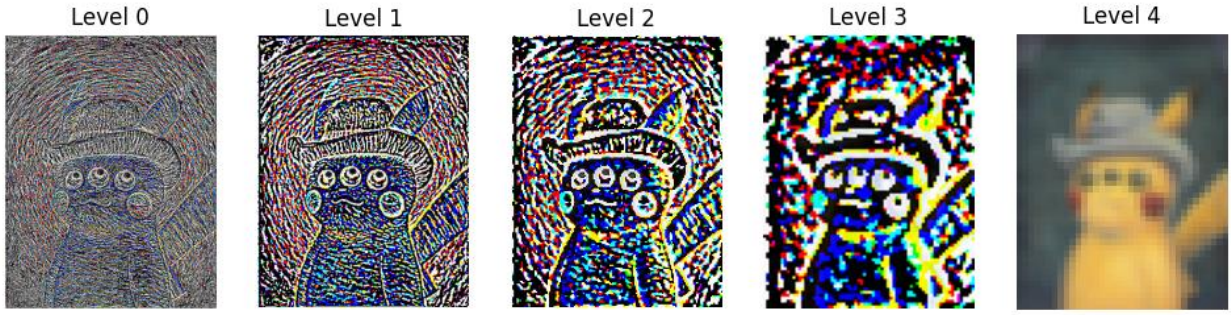
Laplacian 2



Gaussian for Mask



Blended Pyramid



Apple And Orange

Input Image

Image to get a mask

Image1

Mask for Blending

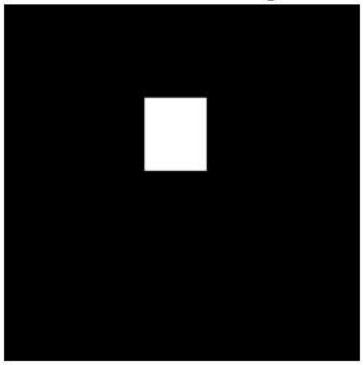
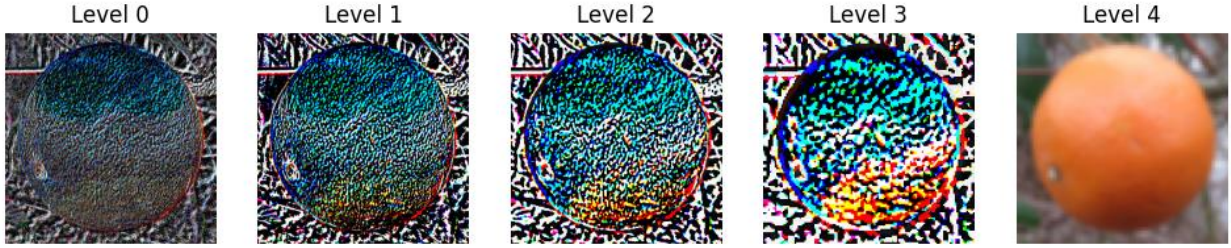


Image2

Blended Image

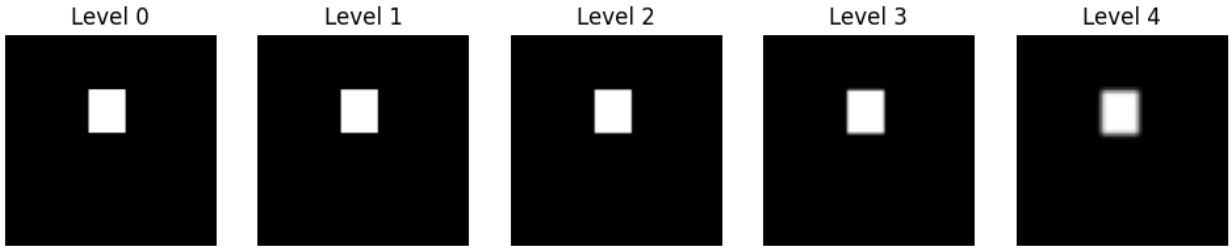
Just a quick clarification here, my implementation doesn't apply mask on a circle area, I only choose rectangle area to apply mask so transition on the edges might be little

Laplacian 1



Laplacian 2

Gaussian of Mask

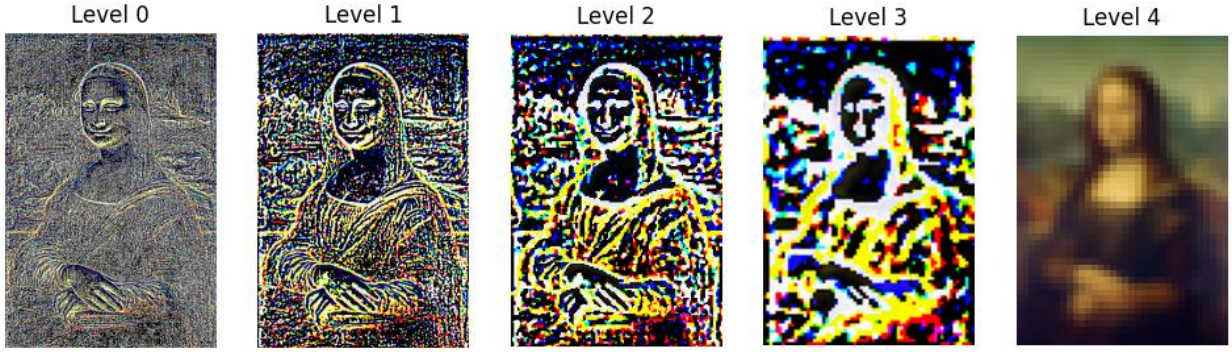


Blended Pyramid

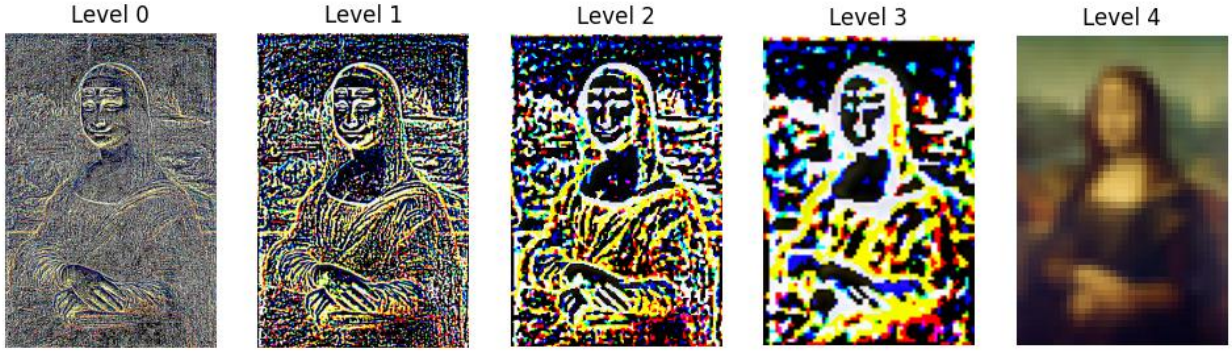
Mona Lisa

Image 2 is the replacement of the mask since I used the same image here

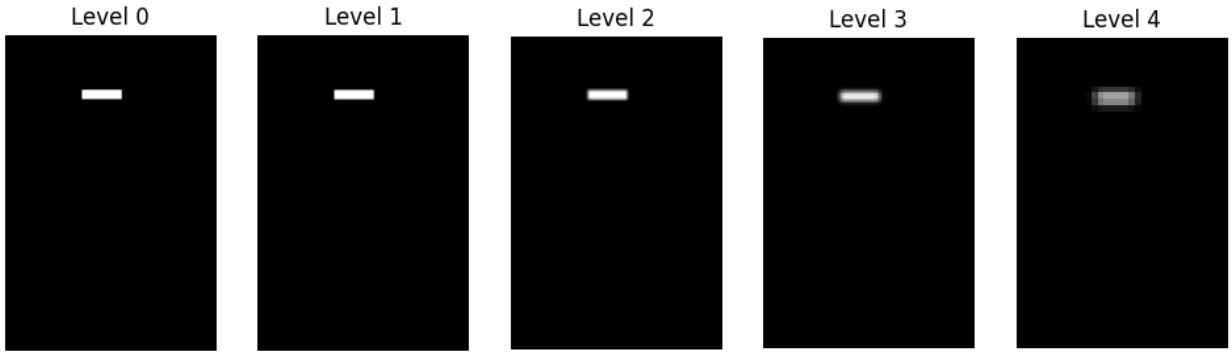
Laplacian 1



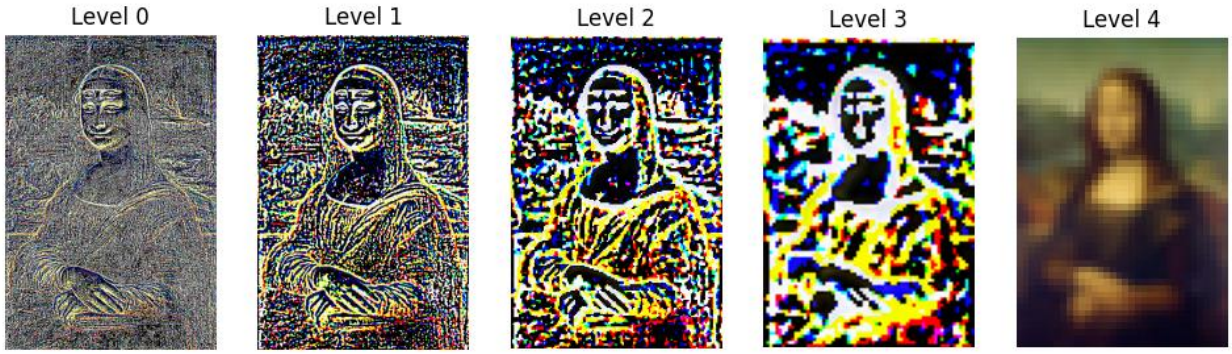
Laplacian 2



Gaussian of Mask

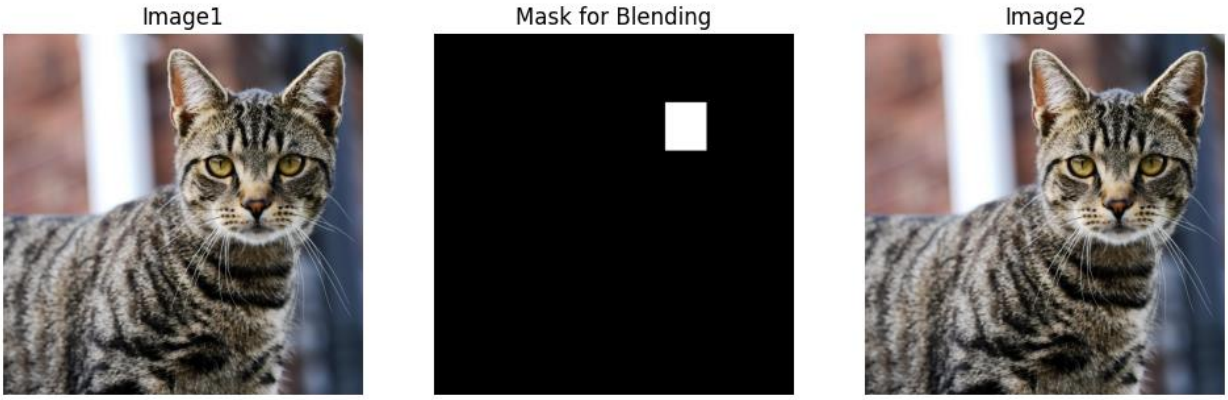


Blended Pyramid



Reconstructed Blended Image

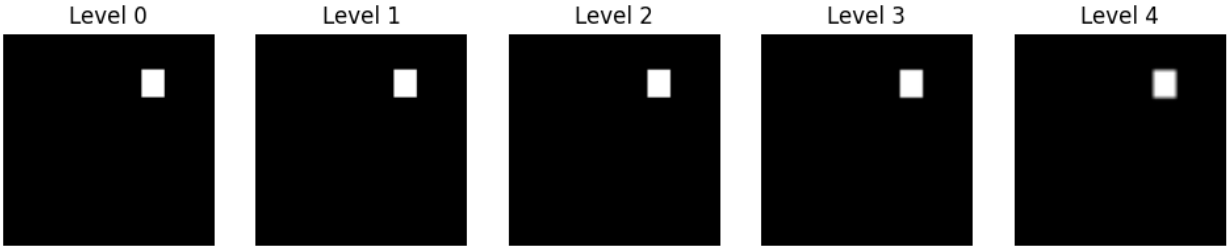
Cat



Laplacian 1

Laplacian 2

Gaussian of Mask



Blended Image

Dog

Image1

Mask for Blending

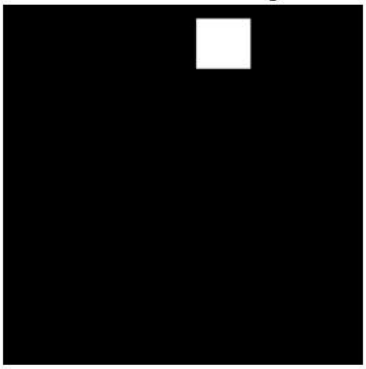
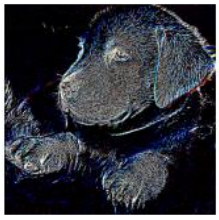


Image2

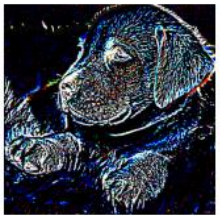
Laplacian 1

Level 0

Level 1



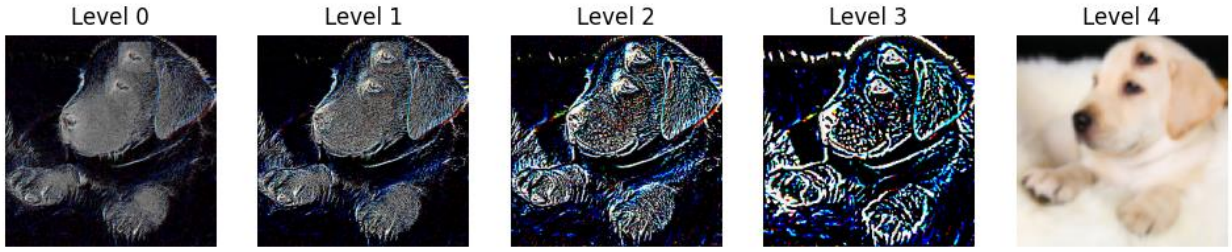
Level 2



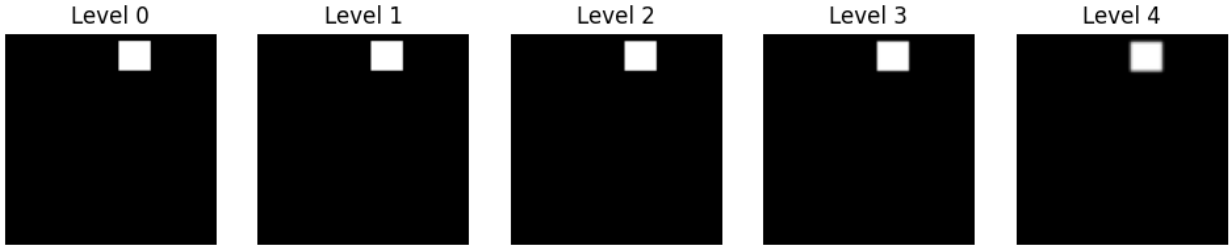
Level 3

Level 4

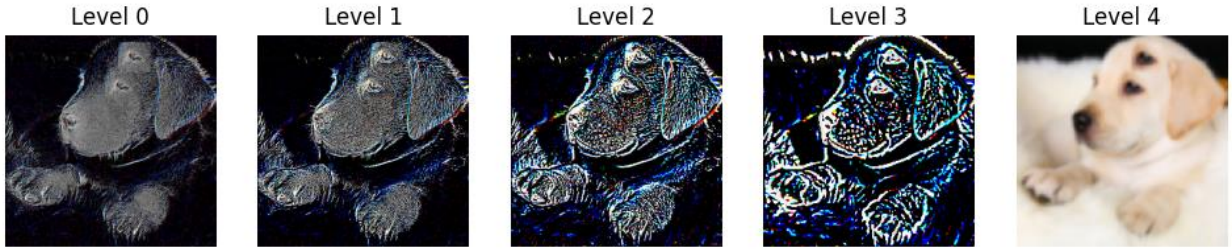
Laplacian 2



Gaussian of Mask

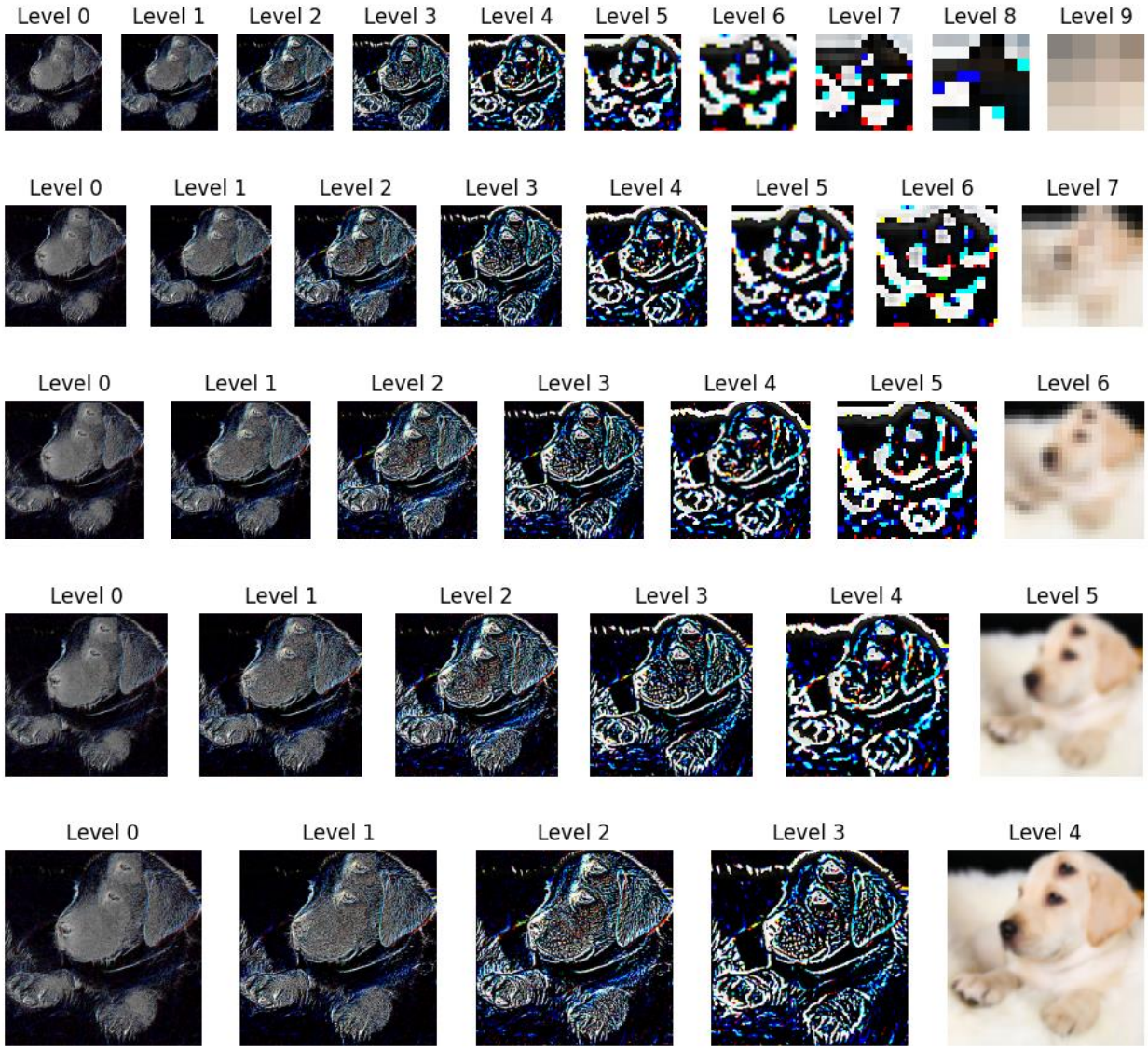


Blended Pyramid



Let’s Increased the Number of Pyramid Levels until the final Pyramid level size is scaled down to minimum dimension.

Blended Pyramids:



*I will add the results to google drive to keep this file low size.

Generally higher level of pyramid level blends more details so it gives smoother image than the lower level of pyramid level.