Addis Ababa Science and Technology University

Collage of Electrical and Mechanical Engineering

Department of Electrical and Computer Engineering

Computer Engineering Stream

Computer Vision

Assignment 1: On High Quality Image Resampling and Image Blending

---

By

1. **Sileshi Nibret** **GSR 217/12**
2. **Gebeyaw Tigabu** **GSR 210/12**

Submitted to: Dr. Beakal Gizachew

February 4, 2021

Part 1: High-quality Image Resampling

Introduction

Image resampling is the process of enlarging or decimation of an image so that the image information at different scales is explicitly available and details of the images are precepted easily. However, in decimating or down sampling process Aliasing effect of signal processing is the main factor for image quality reduction. Aliasing arises when a signal is sampled at a rate that is insufficient to capture the changes in the signal. To overcome this effect smoothing an image using filters like Gaussian or median filtering algorithms before resampling is the good approach. Filters are implemented using weighted sum of pixels in successive window.

Interpolation is a basic technique used extensively to zooming tasks that the input small size image is largen by interpolating the missing pixels based on the existing pixel value. Image interpolation works in two directions, and tries to achieve a best approximation of a pixel's intensity based on the values at surrounding pixels.

Decimation or down sampling is the reduction in spatial resolution while keeping the same two-dimensional (2D) representation. However, in decimating an image smoothing algorithms are very important. In this assignment we have used two smoothing filter (i.e., Gaussian and Median filters) before decimating the size of an image by half, quarter and one-eighth.

Procedures followed

Filters used: Gaussian and Median filters are used for resampling of images

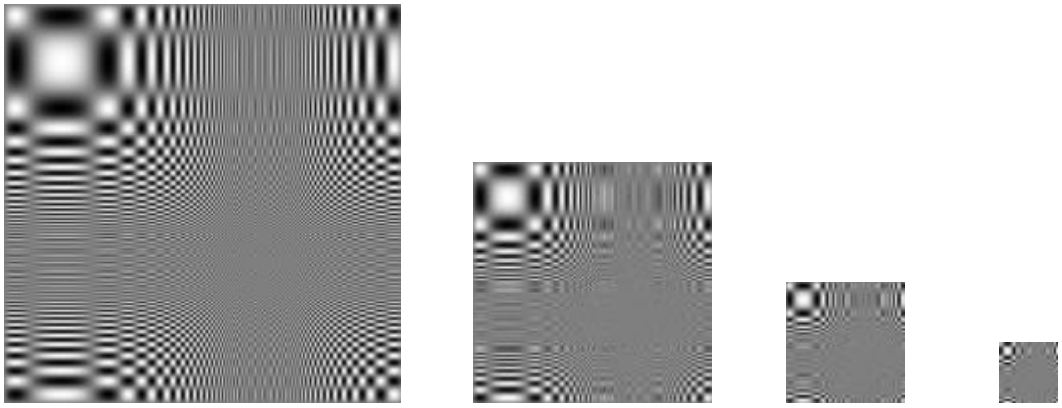
Language used: Python 3.8 with OpenCV framework

Gaussian filter with odd (3,5, and 7) kernel size with different sigma value is applied to the input image.

Sigma =0.5 sigma=1 sigma=1.5

Minifing (Down sampling)

After smoothing the images we have resize the image by a factor of 1/2, 1/4 and 1/8. Since we have smothed the image we can get the information of the image even it is downsampled from Gaussian blur of 7x7 matrix kernel.



Original 1/2 1/4 1/8

The above images are the downsampled images of synthetic chirp image using Median filter. (we can't show all the images output of both filters here because of page number limitation)

Magnifying (Interpolating)

The following images are interpolated using cubic interpolation technique after smoothed using gaussian filter and decimated as above

Interpolated from ½ Interpolated from ¼ Interpolated from 1/8

N.B: All the images in the exercise are included in the working directory and tested by inserting each image

Comparison of Gaussian filter and Median Filter

Median filter is one of the most popular and efficient filters which is simple to implement. The main basic drawback of Median Filtering is blurring the image in process; it could preserve the edges while suppressing the noise as well. Median filters are considered when Gaussian filters can't solve the problem.

This filter is computationally in expensive because of and Median filter further exclude noise and average neighboring pixels but it loses a lot of image-structure information and image details.

Gaussian filter is a particular filter known for blurring and suppressing the noise. This filter is a 2-D convolution operator with the weights selected pursuant to the shape of Gaussian function. This filter is computationally efficient relative to other filters. The degree of smoothing is determined by the given standard deviation σ, very effective to filtering Gaussian noise (low-pass filter) and essential when down-sampling images (Gaussian Pyramid).

Generally, when we consider speed of operation median filter operates faster than Gaussian filter but the smoothing quality for down sampling of an image Gaussian filter is better. Therefore, quality of resampling can compensate the speed of operation.

Part 2: Laplacian Pyramid Blending

Image blending is one of the application areas of Laplacian and Gaussian pyramid. To create a blend of each images are composed into Laplacian pyramid and masking image is decomposed into Gaussian pyramid. An image pyramid is a collection of images, which arise from one source i.e., one original image. In order to create a pyramid, we need to down sample the source image until some desired stopping point is reached. In this experiment we write a python code to accept two colored images and binary mask image to blend the two-colored images based on the mask images Region of Interest.

Procedures Followed

To blend images, we followed the following 4 steps. For each step we have written functions to perform the task.

Step 1: Constructing Laplacian pyramid of each input image

Since the Laplacian image is the result of Gaussian pyramid, we first constructed Gaussian pyramid for the images using the following own defined functions

```
gaussian_pyr_1 = Gaussian_Pyramid(image_1, pyramid_level)   #written function to construct GP
```

laplacian_pyr_1 = Laplacian_Pyramid(gaussian_pyr_1) #written function to construct LP using GP

Step 2: Construct the Gaussian pyramid for the two mask images

mask_pyr_final = Gaussian_Pyramid(mask_image, pyramid_level)#Gaussian Pyramid For mask

mask_pyr_final_compl = Gaussian_Pyramid(mask_image_complement, pyramid_level) #GP for comp

Step 3: Multiply each Laplacian image by its corresponding mask and sum the images

This step is the blending step using the following mathematical operation

*Blended_Image = Laplacian_of_image_a * mask + Laplacian_of_image_B * (1.0 - mask)*

Step 4: Reconstruct the final image from the blended Laplacian pyramid

In this process the blended image is reconstructed from the above Laplacian blended layers of the pyramid so that as to get the blended image for visualization using the following self-written method.

final = Reconstructing_Blended_Image(Laplacian_Added)

In step 3 of the experiment weighted summation is done that is each pixel of the image in levels of the pyramid is multiplied and summed to the other image and mask image iteratively. Since the mathematical formula can be expressed as:

$$\mathbf{SL_l}\,(\mathbf{i},\mathbf{j}) = \mathbf{M_{gl}}\,(\mathbf{i},\mathbf{j})\mathbf{AL_l}\,(\mathbf{i},\mathbf{j}) + (\mathbf{1} - \mathbf{M_{gl}}\,(\mathbf{i},\mathbf{j}))\mathbf{BL_l}\,(\mathbf{i},\mathbf{j})$$

Where: $\mathbf{SL_l}$ is the Laplacian result at *l th* level of the pyramid, $\mathbf{M_{gl}}$ is the Gaussian transformed result of mask image and $\mathbf{AL_l}$ and $\mathbf{BL_l}$ are Laplacian transformed results of the two input images.

Image blending can be done using `cv2.addWeighted()` in OpenCv to sum the weight but in the experiment, we used a for loop operation to keep truck of the weight for renormalization of the image's pixel in the Laplacian pyramid levels.

Therefore, in this stage the total weight of the operation must keep truck of the weight for renormalization at each stage of the Laplacian sum.