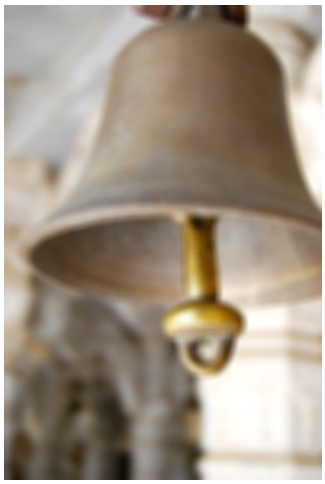
DIP: Assignment 2

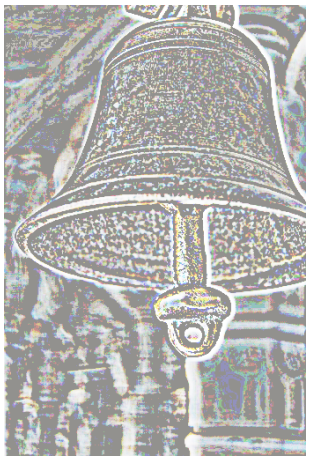
(1) Apply highboost filtering on the image bell.jpg, compare results by varying window sizes (for smoothing filter) and k (the weight factor). Illustrate the steps.

Image blur



Smooth – original

Hist Equalized



Next we add different scaled values(K) of the histogram equalized image back to original to get our highboosted image. As can be seen below the size of the larger the value of the window size, the more details that are amplified. The value K tells us how much the white light is amplified.

Window Size - 5x5

K = 10

K = 50

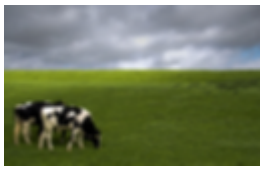
K = 100

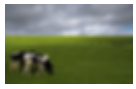
Window Size – 10x10



(2)

(a) Write functions to build Gaussian Pyramids and Laplacian Pyramids from an input image (two different functions). Your function should take as input the image and the number of levels required and should output the pyramids.

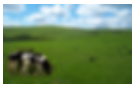





(b) Write a function to reconstruct your image back from the Laplacian Pyramid

Since the laplacian pyramid contains negative numbers, unless scaled produce distored representations like the following. In our usecase, we don’t need to scale them and can use the numbers as they are.

(c) Now use the build and reconstruct functions of Laplacian Pyramids to blend the cows from the image (source.jpg) into target image (target.jpg) using the mask image (mask.png). Observe the results with varying number of levels (from 2 levels to 4 levels of the pyramid).



More the levels the higher the precision with which the images are blended. Below is the image reconstructed with 3 levels. As you'll notice, there is a clear outline around the blended zone. Which is not there in the above images(L=5). Hence more the levels the better the blending effect.

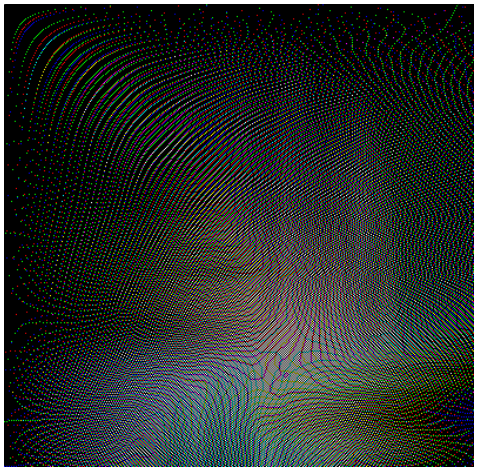
(3) Take *your* RGB face image with a non-plain background and do the following. (Do not use any inbuilt functions)

(a) Compute integral image from face image. Show the 8 bit integral image in the report.

Following is the 8-bit image of the integral image. This image was obtained by converting the integral image from double to uint8. We scaled each channel(RGB) by the max value of that channel and then multiplied by 255.

(b) Reconstruct face image back from the integral image obtained from part(a) (without using loops/recursion).

Reconstruction from the uint8 image produces undesirable lines all over the image.



But using the double image as is, produces a nearly perfect reconstruction

(c) Write your observations on difference between original image and the reconstructed image. Can you think of the various applications where we can use integral image (except from object/face detection)?

As can be seen plainly from the uint8 reconstruction a lot of noise is induced due to the reconstruction due to loss in precision. Similarly, we can observe a little blurring in the case of the double reconstructed image. The integral image can be used a mechanism to calculate the variance of an image. It can also be used represent higher dimensional images accurately.

(d) How many addition operations are there in your implementation ? Can you reduce the number of addition operations by half (2MN where MxN is the size of your face image)? Does it also affect storage, if yes please write your observations.

My original algorithm runs by running in two passes. It first computes an integral image in the X-axis, then followed by the Y-axis. This reduces the number of operations per 2x2 block by a single operation and results in a much faster implementation for calculating the integral image. The current implementation uses 2*M*N addition operations.

(4) We discussed the edge preserving bilateral filter in the class. A detailed description with equations can be found at the wikipedia page ((https://en.wikipedia.org/wiki/Bilateral_filter)). Attempt the following:

a) Implement an 5 × 5 bilateral filter and apply it to the gray scale image "face.png" (Note: be careful in choosing the value of σ).

The result of bilateral filtering can be observed below. Notice that all wrinkles have been smooth, while the edges of various facial features have been enhanced.

b) The filter can be extended to color images by simply applying the filter to each color channel separately. Use the image "boy-smiling.jpg" to test color image bilateral filtering with different window sizes (5 × 5; 10 × 10 and 15 × 15) and different values of σ. Comment on the effect of changing window sizes and σ.

Here is the image for the boy against the original.

As you can notice, in the first image, his freckles have been smooth.

The higher values of sigma-r that is the variable effecting the intensity difference, works inversely. Hence the smaller value of sigma-r results in sharper edge magnification. Small values for sigma-d result in the pixel values which are far apart being clubbed together for intesity considerations. The images below illustrate the differences. The left most image is the original image. The one in the middle is an image which has been bilateral filtered with high sigma-d and low sigma-r(High edge enhancements). The rightmost image is obtained by setting the value of Sigma-d and Sigma-r both high, giving a similar effect as the gaussian blur.

Original

High sigma-d, Low sigma-r

High sigmad, High sigmar

c.) Does it makes sense to develop an inverse bilateral filter, which blurs an image at edges and preserves the homogenous regions. If it makes sense, design an inverse bilateral filter and suggest its applications.

By definition, a inverse bilateral filter would be one that explicitly blurs well defined edges.

This can help identify regions on the image which may be bounded by these edges.

(5) (a) Write a function to perform image warping given a transformation function.

In this assignment I used the imtransform function to implement such a generic version of these operations. Below are the images of the same image being translated, rotated, and sheared.

(b) Use it to implement the Twirl transform given in the lecture slides.

This operation is shown below. It was built by composing a function using the polar coordinates and a backward mapping of pixels to obtain.