CS 682 Assignment 1

Venkata Sasank Mudigonda

***Abstract*— This assignment summarizes my work on various filtering operations including Fourier transforms, up-sampling, down-sampling, image gradients etc.**

I. CONVOLUTION WITH FILTERS

The convolution operation comes from the domain of signal processing and helps in analyzing and processing signal. It can also be applied to image processing by converting images into their corresponding individual signal constituents. Below is a 2D discrete convolution operation where f is a filter and g is the image we want to perform the convolution operation on.

$$(f * g)(x, y) = \sum_{i,j} f(i, j)g(x - i, y - j)dy$$

Given filters are: $f_a = \frac{1}{9}\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix}$ $f_b = \frac{1}{3}\begin{pmatrix}0&0&0\\1&1&1\\0&0&0\end{pmatrix}$ $f_c = \frac{1}{6}\begin{pmatrix}-1&0&1\\-1&0&1\\-1&0&1\end{pmatrix}$ $f_d = \frac{1}{3}\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}$. f_a is called a *box filter* and f_c is sensitive *vertical edges*. *Scipy* is used to implement the convolution operation. Among the above filters, f_a, f_b, f_c are *linearly separable* but not f_d because it's an identity matrix. $f_a = \frac{1}{9}\begin{bmatrix}1\\1\\1\end{bmatrix}\begin{bmatrix}1&1&1\end{bmatrix}$ $f_b = \frac{1}{3}\begin{bmatrix}0\\1\\0\end{bmatrix}\begin{bmatrix}1&1&1\end{bmatrix}$ $f_c = \frac{1}{6}\begin{bmatrix}1\\1\\1\end{bmatrix}\begin{bmatrix}-1&0&1\end{bmatrix}$. Fig. 1 shows how an image is transformed after performing convolution operation with all the above filters

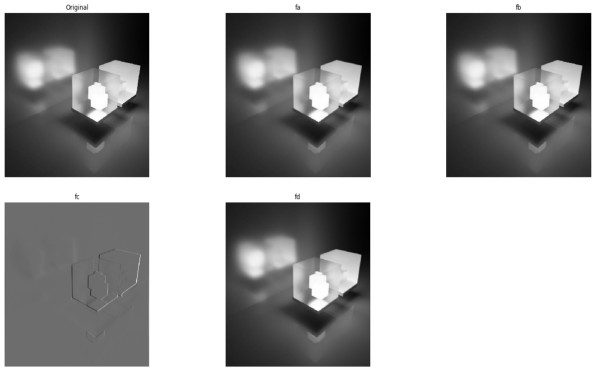

Fig. 1: Effects of the filters f_a, f_b, f_c, f_d on an image

II. IMAGE DERIVATIVES

I_x and I_y represent the derivatives of image in horizontal and vertical directions respectively. They are computed by performing convolution operation on the image with horizontal and vertical Sobel filters (represented by S_x and S_y) with respect to their corresponding directions, x and y.

$$\frac{\partial I}{\partial x} = S_x * I \qquad \frac{\partial I}{\partial y} = S_y * I$$

$$Direction \quad \theta = \tan^- 1(\frac{I_y}{I_x})$$

$$Amplitude \quad \|\nabla I\| = \sqrt{I_x^2 + I_y^2}\tan^- 1(\frac{I_y}{I_x})$$

$$Laplacian \quad \nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

All the above metrics are depicted in Fig. 2 and Fig. 3 with *PiYG* color map. The whiter regions in the Laplacian is where *zero-crossing* happens. The image depicting the amplitude of the derivative has whiter regions along the edges because changes in intensity values along the edges are might higher.

Fig. 2: Image metrics from sample image 1

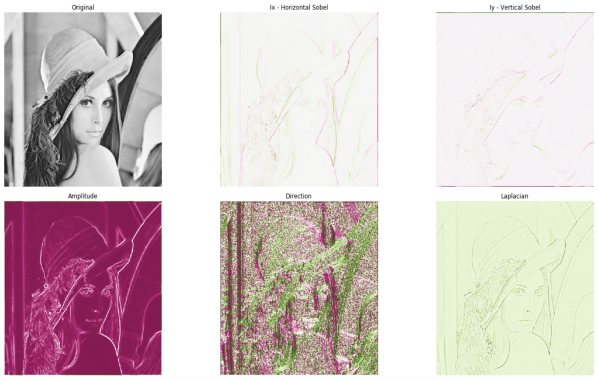

Fig. 3: Image metrics from sample image 2

III. GAUSSIAN FILTERING

A. GAUSSIAN FILTERS

The images in Fig. 4 show the distribution of the Gaussian filters with of size 31x31 with different standard deviations σ of 7 and 21 respectively. The first filter's σ obeys the convention that filter size must be more than 3σ. This is because for all the practical reasons the values of the function beyond 3σ from origin is very close to 0. So, if we keep the filter sizes greater than 3σ, then whole Gaussian bell is taken into account and at the filter's edges your weights will asymptotically tend to zero. We can clearly see the fall of the amplitudes around the edges of the filter. But for the filter with $\sigma = 21$, the distribution is not completely collapsed around the edges of the filter. Therefore we're actually missing out on a filter's full capacity to act as a smoothing filter.

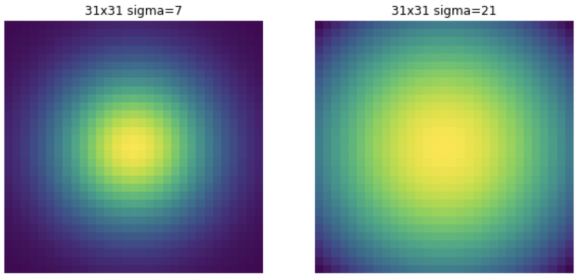

Fig. 4: The plot of the Gaussian distribution

B. APPLYING GAUSSIAN FILTERS TO IMAGES

Various filters of different sizes and σs have been applied to two sample images as shown in Fig. 5. As the filter sizes and σs increase, so did the blur aspect on the images.

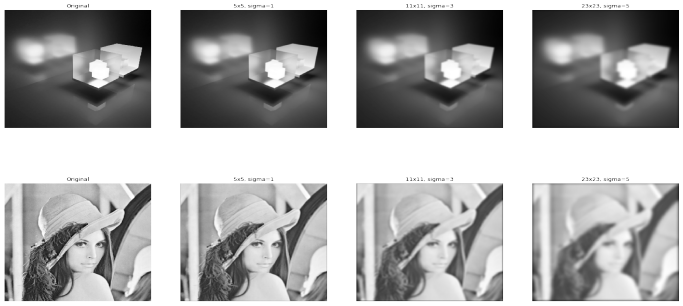

Fig. 5: Gaussian Kernels

IV. DERIVATIVE OF CONVOLUTION

$$\frac{\partial(f * I)}{\partial x} = \frac{\partial f}{\partial x} * I$$

The above equation shows how te derivative of the convolution can be simplified by taking the derivative of just the filter. Fig. 6 shows the results is not exactly what's expected from the above equation. The RHS of the equation has greater contracts than the LHS.

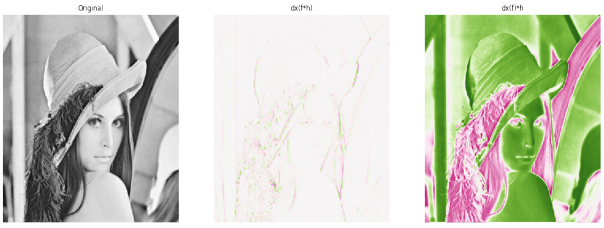

Fig. 6: Derivative of convolution

V. IMAGE UP-SAMPLING

A. BASIC KERNELS

The results from Fig. 7 shows the up-sampling outputs of various the following interpolation strategies: *nearest neighbor, bilinear, and bicubic*. Each one is more complicated than the previous one and better estimate the underlying the distribution of the image to interpolate. This is clearly observed from the results in Fig. 7.

Fig. 7: Up-sample Kernels

B. FOURIER TRANSFORMS

Fourier transforms can also be used to up-sample the images. It essentially follows the steps below.

1) Apply Fourier transform to the image
2) Perform *zero-padding* on image to be up-sampled and bring it to the desired (target) size (this hapenning in the frequency domain)
3) Apply Inverse Fourier transform to the padded image and bring it back to the spatial domain. When the Fourier transform is applied to the image using $np.fft.fft2$, most of the mass of the frequency space is located in the corner of the image (see Fig. 8, second image). We need to use $np.fft.fftshift$ to bring the mass of the Fourier transform to the center of the image. We use $np.fft.ifft2$ to apply the Inverse Fourier transform. The results in Fig. 9 show that up-sampling with Fourier transform is much more smoother than the previous kernels.

VI. HYBRID IMAGES

A hybrid image is formed by applying a low-pass Gaussian filter to one image and high-pass Gaussian filter is applied to another, and added to form resultant image. This way we capture features from both the images and create a new image with both of the source image features. In Fig. 10, we combine the images of Einstein and Lenna to create a

Fig. 8: Fourier Shift

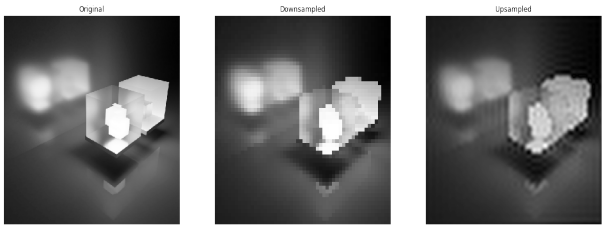

Fig. 9: Fourier Up-sampling

Fig. 10: Hybrid

different personality. In Fig. 11, we utilize the Gaussian blur filter to down-scale the hybrid image at various scales to form a *Gaussian Pyramid*.

Fig. 11: Gaussian Pyramid