Lab 4
Image Filtering

Jonathan Ananda Nusantara
jan265

In this lab, Image filtering will be explored in both the frequency and spatial domain. These image filtering will be performed using VisionX commands. Two of the same images will be used, and as a result a fair filtering performance comparison between the two domains can be done.

A total of 2 images are filtered, which are ted.vx and sshtl.vx, shown in Figure 1 and 2 below, displayed in png format:

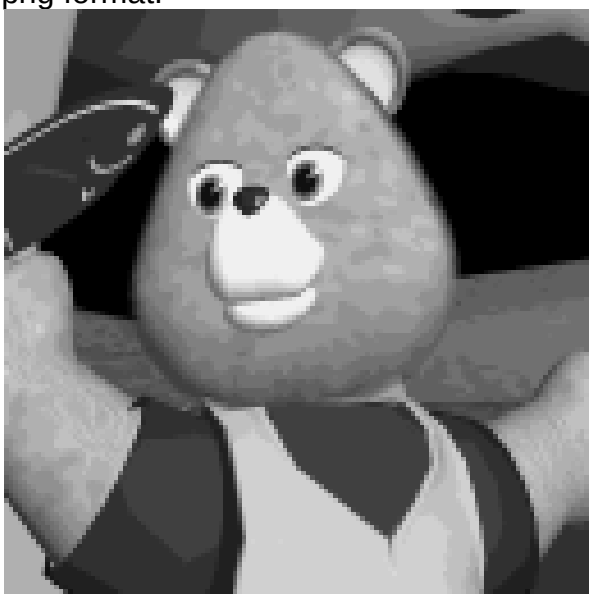

Figure 1 – Ted.vx image for filtering experimentation

Figure 2 – Sshtl.vx image for filtering experimentation

Frequency Domain:

The experiment starts by using the ted.vx image in Figure 1. The first step would be to convert the image from spatial domain to frequency domain, by using FFT. This can be done by using the following command in terminal:

vfix -float if=ted | vfft of=ted.fft

The “vfix” is a VisionX command to change the data type of the image, and ted image is used as input to transform it to have a float data type. Then, the output image is send to be an input to the “vfft” command in VisionX, which is the command to perform FFT (Fast Fourier Transform) on the image. Thus, the output image “ted.fft” would be the image in the Frequency Domain. In the manual of “vfft” command, it says that the output file will be a 2-channel file. This is expected because in Frequency Domain, there are the Real and Imaginary portion. Notice the “|” symbol that is used in the command, which is called a “pipe”. This will allow us to redirect an output of one command to another.

Q: What are the two channels in the output of “vfft”?
A: Real and Imaginary channel.

Since ted.fft is a 2 channel float image, vview cannot directly display the image. The script “vexfft” that was provided with for the lab can be used to extract the viewable portion of the FFT transform of an image. The magnitude and phase of the image need to be extracted, which can be done by passing “-m” or “-p” flag when running the “vexfft” script. The followings are executed to create these viewable FFT images:

sh vexfft ted.fft -m of=ted.mag
sh vexfft ted.fft -p of=ted.phase

With the above commands executed, 2 images are created, which are “ted.mag” and “ted.phase”, which are the magnitude and phase of the FFT transform ted image.

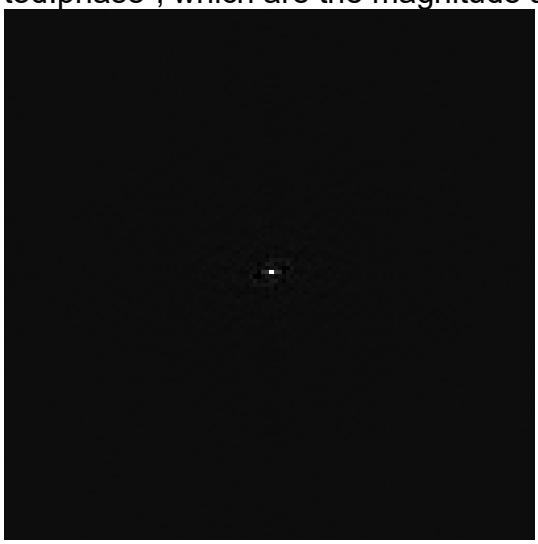

Figure 3 – Magnitude of FFT-transformed ted.vx.

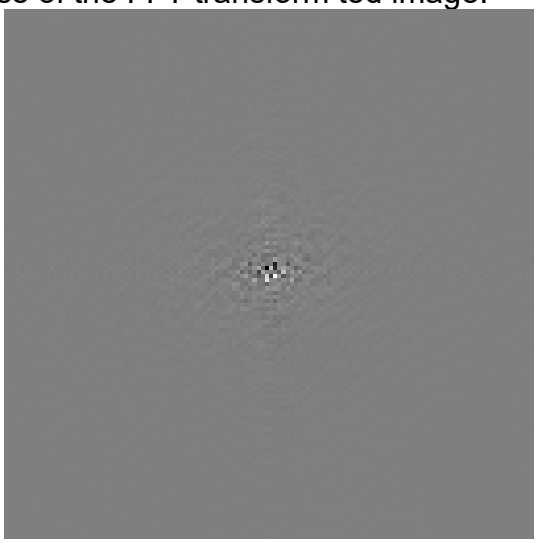

Figure 4 – Phase of FFT-transformed ted.vx.

Some pattern or information can be seen in the phase image shown in Figure 4. However, not much information can be seen from the magnitude image in Figure 3, beside the white dot in the middle. As a result, something needs to be done to perform a better visualization. To better represent the magnitude image, the DC offset of the original image will be removed. One easy technique of doing this is by subtracting each pixel with the mean pixel value of the image.

From “Tools -> Image Statistics”, the mean pixel value is seen to be 110.366. So this time, this mean pixel value will be subtracted from the original image first before

transforming to frequency domain using FFT. This is done by executing the following command:

vfix -float ted | vpix bf=-110.366 | vfft of=ted.xfft

Like before, this command would first transform the original image to have a float data type. Then, the output is passed to the "vpix" command, which would perform a point operation on the image. The "bf" parameter, which would add the defined value to each point, is set to be the negative of the mean pixel value. The output is then passed to "vfft" command which would output the FFT transformed image.

Now that the DC offset has been removed, the log magnitude of the FFT image can be displayed to provide a better visualization. This time, the "-l" flag in the "vexfft" will be used to perform "log" before extracting magnitude information. The following is executed:

vexfft -m -l ted.xfft of=ted.xlmag

The output image "ted.xlmag" significantly provided a better visualization:

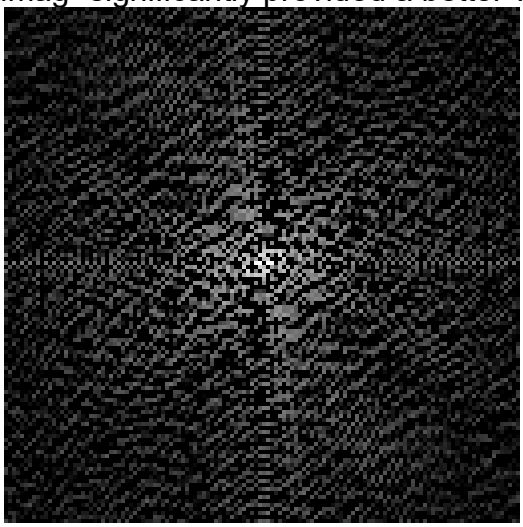

Figure 5 – Log magnitude of FFT-transformed ted.vx, with DC offset removed.

Q: In the above steps how did we manage to visualize the magnitude data?
A: As have been explained above, we manage to visualize the magnitude by removing the DC offset. This can be done by first subtracting the mean pixel value of original image from each pixel, using the "vpix" VisionX command. Then, the FFT is performed on that image. In order to view the image, the "vexfft" script is used to display the log magnitude of the image. From the result in Figure 5, there are much more information that the viewer can learn, such as the noise around the image.

Q: How was the value for the bf= parameter selected?
A: The bf parameter used (-110.366) is coming from the mean pixel value of the ted image, which is 110.366. Since we wanted to remove the DC offset by subtracting each pixel with the mean pixel value, the bf=-110.366 is chosen.

The next step of this experiment is to create the frequency domain filter. The first step is to execute the following:

vgenim x=128 y=128 c=32,32 hi=1 | vfix -float of=f1
vchan if=f1 ig=f1 of=fil

Q: What does the first line accomplish?
A: The “vgenim” command is used to create a simple shape image. They are defined to have 128 pixels across x and y plane. Then, the “c” parameter means that a circle will be created and it will have a radius of 32. The “hi” parameter is used to set the pixel value of the shape, which is set to 1. The background of the image will have pixel value of 0. The output is then passed to “vfix” which would convert the data type of the image to float. The output is named “f1”.

The second step of creating the filter is merge the 1-channel image that was created into a 2-channel image that is needed for the image filtering purpose. The following is executed:

vchan if=f1 ig=f1 of=fil

Q: Why is `vchan` being used?
A: “vchan” is a VisionX command that is used to performed operations on multi-channel image. For our purpose, it will be used to merge two 1-channel images into a single 2-channel image. This is done by passing the same “f1” image in both the input “if” parameter and second image “ig” parameter. The output of it is a 2-channel filter “fil”.

The filter that will be used in frequency domain is displayed below:

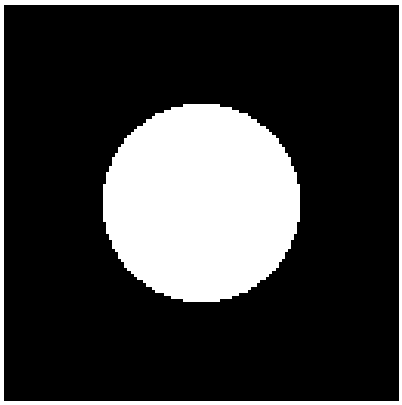

Figure 6 – 2D representation of the “fil” filter.

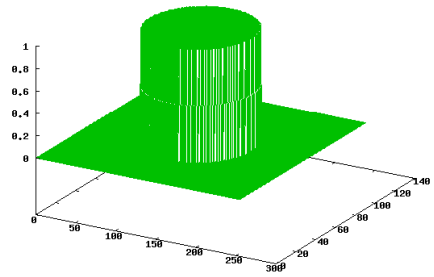

Figure 7 – 3D representation of the “fil” image by using vplot.

The Figure 6 above shows the filter when viewed in vview. It can be seen as a circle form the top. When the “vplot” command is used to display the filter, it looks like a cylinder as shown in Figure 7.

Now that the filter is ready, the FFT-transformed ted image is ready to be filtered. From Figure 6, the filter looks like a Low Pass Filter. This will be proven in the output. In frequency domain, filtering would be done by multiplying the image with the filter. The following command is executed to perform the filtering:

vop -mul if=ted.fft ig=fil of=tedf.fft

The “vop” command will combine images based on the chosen flag. The our purpose, the “-mul” flag is chosen to multiply the two images passed. The output of the command is the “tedf.fft”. In order to view the image, the script “vexfft” was executed again to get the magnitude and phase:

sh vexfft tedf.fft -m –l of=tedf.mag
sh vexfft tedf.fft -p of=tedf.phase

The resulting images from above is as below:

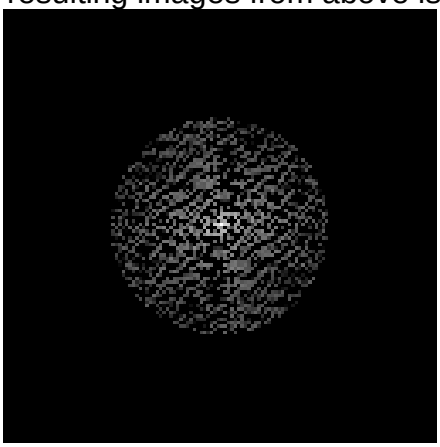

Figure 8 – Magnitude of tedf.fft filtered image

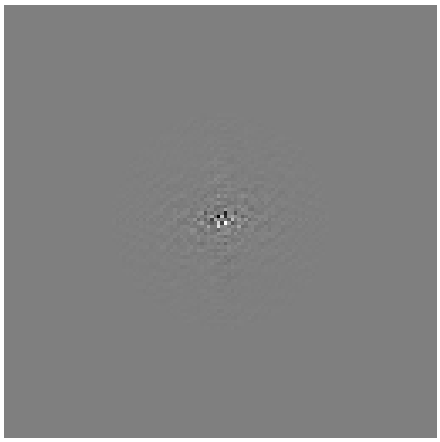

Figure 9 - Phase of tedf.fft filtered image.

When compared to the unfiltered image in Figure 4 and 5, the effects of the filter can be seen in Figure 8 and 9. The filter, which is in the shape of a circle, will remove the noise that is outside of the boundary of the circle. As a result, when multiplied with the original image, the pixels outside the circle will be removed. This can be seen much more clearly in the magnitude, and is a little more difficult to see in the phase. Now that the effects of the filter has been proven as Low Pass Filter, which removes the high frequency outside of the boundary of the filter, which is usually the noise, the filtered image will be transformed back to the spatial domain. The “vfft” command is again used to perform an inverse FFT:

vfft -i if=tedf.fft of=ted.f

Notice the “-i” flag used which means that it will be inversed. The output image “ted.f” is a spatial domain image that has been filtered. Interestingly, this output image was not able to be displayed in vview, so the “vexfft” is again used to extract the magnitude of the image. The resulting image is shown below:

Figure 10 – Side by side comparison between original ted image (on the left) and filtered ted image (on the right)

From Figure 10 above, it can be confirmed that the filter used is a Low Pass Filter. The low pass filter will blur the image, which is clearly seen in the picture on the right that is much more blur compared to the image on the left. The goal is to remove the noise in the original image. Some noise exists on the ground in the original image (located below the mouth level of the bear). After filtering, some of the noise has been removed. Another significant difference is the smoothing of the edges. In the original image, the edge of the sleeve and the arm of the bear is clearly defined. However, the edges are smoothed up in the filtered image and is not clearly defined. This shows the effects of low pass filter on edges.

Now that the ted image has been filtered, the same operation is done on the sshtl image in Figure 2. The image is first transformed to frequency domain using “vfft” command. The resulting magnitude and phase images of the transformed image are as below:

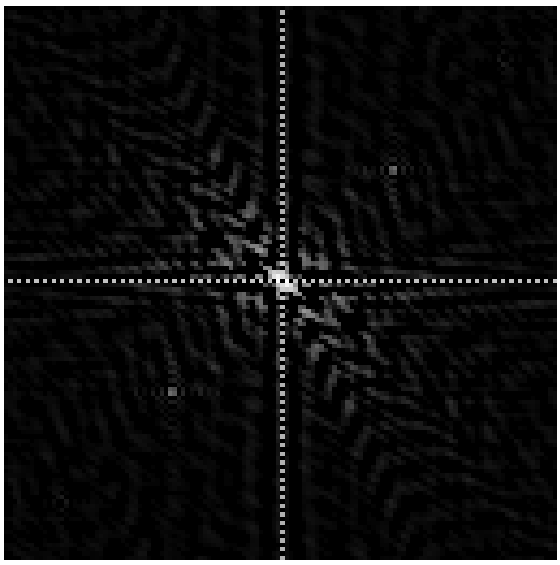

Figure 11 – Log magnitude of FFT-transformed sshtl image.

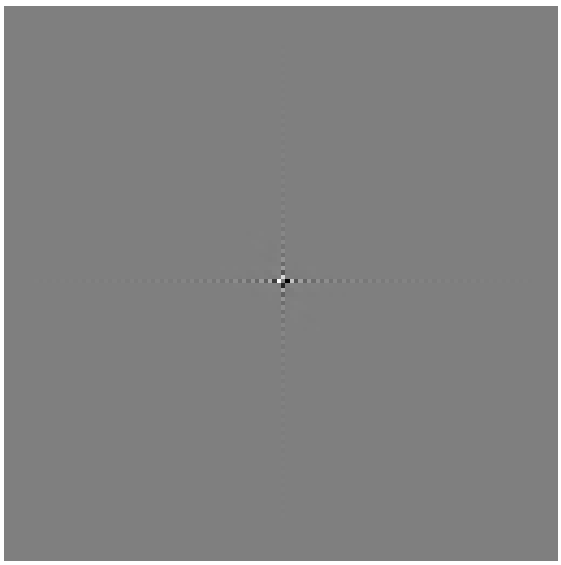

Figure 12 – Phase of FFT-transformed sshtl image.

For visualization purposes, the “log” operation is done before scaling in extracting the magnitude of the image using the “vexfft” script, shown in Figure 11. From the two figures above, some information can be interpreted. For example, in Figure 11, there are patterns across the entire image.

Next, the FFT-transformed image is multiplied with the filter using the “vop” command. The “vexfft” script is again used to display the magnitude and phase of the resulting filtered image:

Figure 13 – Log magnitude of filtered sshtl image in frequency domain.

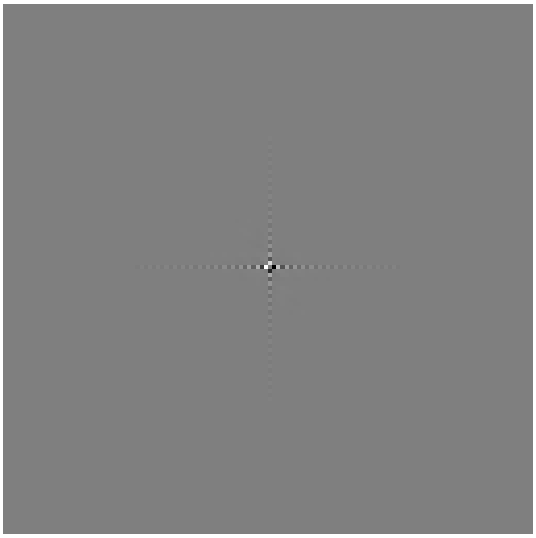

Figure 14 – Phase of filtered sshtl image in frequency domain

From the filtered magnitude and phase images shown in Figure 13 and 14, the effects of the filter can again be seen. Since the filter has a shape of a circle, after the multiplication the information outside the circle are zeroed out. In the magnitude, this means that the high frequencies are removed and noise are usually high frequency. A similar effect can be seen in Phase image, although it is harder to see.

The frequency-domain filtered image is then converted back to spatial domain by performing inverse Fast Fourier Transform using “vfft” with “-i” flag. The resulting filtered image, after extracting the magnitude using “vexfft” is shown below:

Figure 15 – Side-by-side comparison of original sshtl image on the left and filtered sshtl image on the right.

Comparing the two images above, the effect of the LPF on the edges can be seen. In the original image, the edges of the shuttle can easily be perceived by human eye. On the other hand, the edges were smoothed up in the filtered image and it is

difficult to tell now. What was interesting is that how the background in filtered image now has tiny squares, which is the result of removing the high frequencies form the original image.

Spatial Domain:

In this next section, we will be filtering the ted image in the spatial domain. This means that it will not involve transforming the image using Fast Fourier Transform. The goal is to see the results of a filtered image in the spatial domain and comparing it with its frequency domain counterpart.

The first step is to create the filter in the spatial domain. The “vgenim” command is again used to create a simple shape image:

vgenim r=2,2 x=10 y=10 hi=1 of=sf1

The size of the image is of 10 by 10 pixels in the x and y plane. The “r” command is used to create a rectangle of size 4 by 4, which is a square. The way the “r” flag works is that half of the length is defined, so 2 is half of 4. The “hi” parameter means that the rectangle will have a pixel of 1, while the background pixels are 0. Next, the following command was executed:

vconv if=sf1 k=sf1 | vfix -float of=sf2

The “vconv” command is used to convolve two input images, with both images being the “sf1” output of a 4 by 4 rectangle. The output is then converted into a float data type using “vfix” command. The result filter is a triangle, since two squares are convolved with each other.

However, one problem still exists. Since filtering a image in spatial domain is done by convolving an image with a filter, there exist a possibility where the output image pixel range may go beyond 0-255 range. Thus, the filter needs to be scaled down to make sure that the output image stays within the 0-255 range. This can be done by dividing each pixel in the filter with the multiplication of total number of pixels in the filter and the mean pixel value in the filter. Firstly, the following command is executed:

vps sf2

This command would display the image statistics. The important statistics value that will be used is the total number of pixels, whose value is 100, and the mean pixel value, which is 2.56. Next, the “vpix” is used to perform a point operation to scale down each pixel value in the filter:

vpix if=sf2 tf=0.0039 of=sfil

The “tf” flag, whose value will be multiplied with each pixel, is set to 0.0039. This value is equivalent to 1 / (100*2.56). The resulting output “sfil” is a triangle with a scaled down pixel value. The filter is displayed using the “vplot” command:

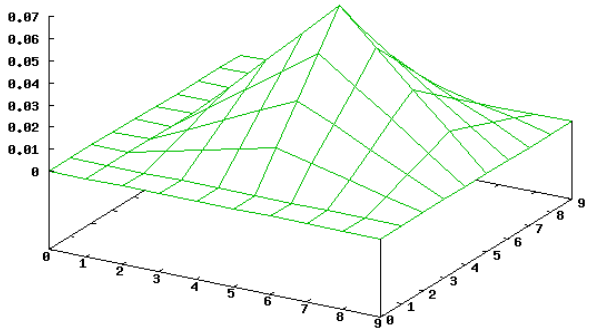

Figure 16 – Spatial low pass filter (triangle) displayed using vplot.

From Figure 16, the filter is a triangle and it is a low pass filter. It will smooth up every pixel by performing a weighted average of the pixel itself and its neighbors based on their distance to the currently convolved pixel.

The final step is to finally filter the image. In spatial domain, the image will be convolved with the filter. This is performed using the following command:

vconv if=ted k=sfil of=ted.sf

The “vconv” command is used to convolve the image “ted” with the filter “sfil”. The resulting image is a filtered image that is smoothed and blurred using low pass filter:

Figure 17 – Side-by-side comparison of the original ted image (on the left) and the ted image filtered in the spatial domain (on the right).

The goal of a low pass filter is usually to remove noise. On the filtered image on the right, the noise on the bear’s face, hand, or the ground has now been filtered out. Another effect of the low pass filter is that the image will be blur, which can easily be seen. The edges of the bear or clothes worn by the bear is also much smoother.

The next thing to observe is “ted.f”, the image filtered in frequency domain, and “ted.sf”, the image filtered in the spatial domain, is different to each other:

Figure 18 - side-by-side comparison of the ted image filtered in the frequency domain (on the left) and filtered in the spatial domain (on the right).

From the above image comparison, what is clear is that “ted.sf” on the right is more blur. More of the noise from the original image is also filtered out. The edges throughout the entire image is also less defined. On the other hand, “ted.f” still has noise on the bear’s face that can still be spotted. It also has the tiny squares throughout the image, which reduces the quality of the image.

The differences between the two images is because two different low pass filters are used. The “ted.sf” uses a triangle-shaped filter in the spatial domain, which is a “sinc-squared” function in the frequency domain. This means that there is no cutoff for high frequencies, but they are just scaled down to be really small. On the other hand, “ted.f” uses a cylinder-shaped filter in the frequency domain, which would have a cutoff frequency for high frequencies.

On the other hand, in the spatial domain, “ted.f” cylinder-shaped filter in the frequency domain will look like a sinc function in ths spatial domain. As a result, when ted image is convolved with a sinc function, it makes sense to have a ripple effect as shown on the left image in Figure 18. On the other hand, since “ted.sf” is a ted image convolved with a triangle, it is smoother and does not have the ripple effect. The difference in the shape of the filter used caused a difference in the resulting image. We can choose the right filter to use for different result image that we want.

Spatial version of frequency-domain filter:

The final part of the lab is to printout a 2D plot of the spatial version of fil. Since fil is in the frequency domain, the first thing to do is to transform it to the spatial domain. This is done by the following command:

vfft -i if=fil of=fil-ifft

Using the “vfft” command, an inverse Fast Fourier Transform is done to fil. The output is displayed using “vplot” and it looks like a sinc function:

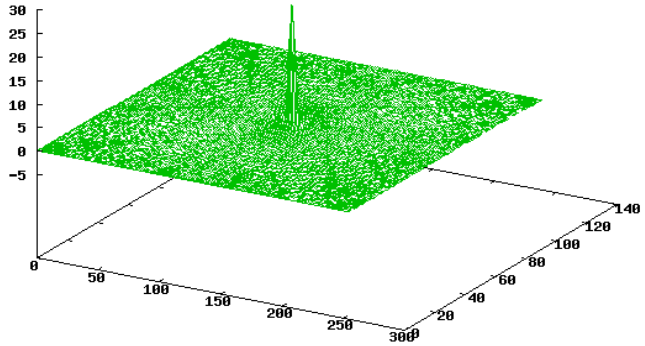

Figure 19 – 3D representation of fil in spatial domain.

The goal is to display the 3D sinc function as a 2D representation. The first step is to take the magnitude of the filter using “vexfft”:

sh vexfft -m fil-ifft of=fil-ifft.mag

The output of that command is a 2D representation of sinc when seen from the top. However since the image is quite dark, the “vpix” command is used to multiply each pixel value by 2, to make all of the pixels brighter/higher intensity. The resulting 2D representation is as below:

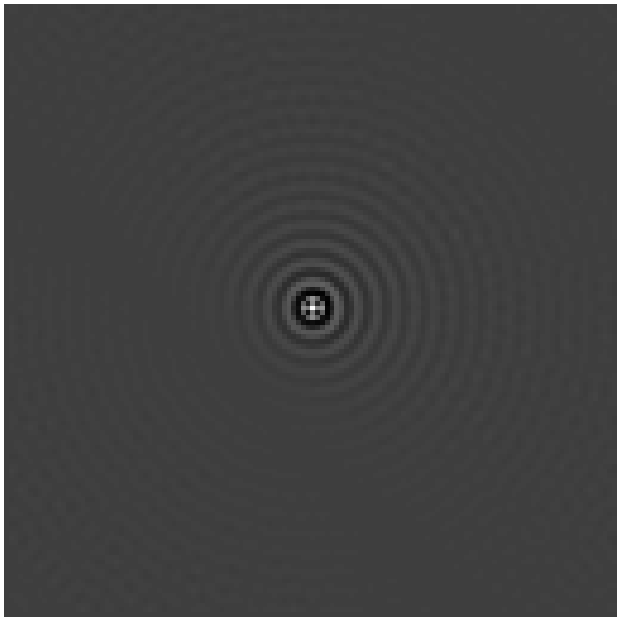

Figure 20 - 2D representation of fil in spatial domain.

Looking at Figure 20, one can see that this is a sinc function. It has the highest peak in the middle of the image. It would then decrease ripple through the entire image starting from the middle to the corners of the image.

For a better depiction of the inverse Fourier transform of the filter fil, which is fil-ifft (shown in Figure 19), we should try bound the more significant portion of the function. This is to focus more on the middle peak of the sinc function. This can be done by utilizing a VisionX command to clip a region of an image, which is “vclip”. The command is called as below:

vclip fil-ifft of=fil-ifft-clip xl=40 yl=40 xh=87 yh=87

Since the original image is a 128 x 128, ranging from 0, 0 to 127, 127. The image is clipped by 40 pixels from both ends, in x and y dimensions. 87 is 127 subtracted by 40, while 40 is 0 added by 40. Then, the magnitude is again extracted using the “vexfft” script. Finally, “vpix” command is used to multiply each pixel by 2 to make the overall image brighter. The resulting image is as below:

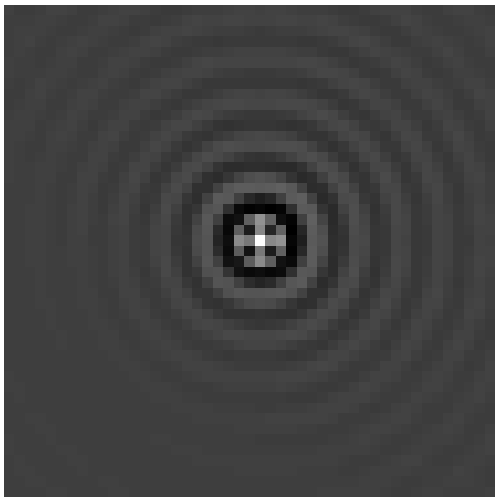

Figure 21 – Clipped version of 2D representation of fil in spatial domain.

The image above is a zoomed in or clipped version of Figure 21. You can see how the middle of the image is the peak, which is the most intense greyscale (highest pixel value. The first black ring around the peak is the darkest, because it is the deepest valley in a sinc function. As it rippled outwards, the difference in intensity between the light and dark ring gets more subtle and this confirms that it is a sinc function.