A Novel Generalised Filter based on Fourier Transform

Naksatra Kumar Bailung
Newcastle University
Newcastle, UK

***Abstract*—With the advancement of wavelet transform and subsequent methods, filters based purely on fourier transform are rare to come across these days. Processing noisy images is also on the rise as display are slowly overcoming the limit of human visual acuity. We propose a novel filter that works in RGB as well as the grayscaled domain and also gives leeway as a n-dimensional filter which preserves most of the information of the image.**

***Index Terms*—fourier filter, nFFT. flower transform**

I. Introduction

The new age of technology has required as well as advanced the study of processing signals. With the only theoretically available method of sampling to be discretized, we heavily rely on discrete signal processing to analyse signals. Signals processing is required in the medical, communication, manufacturing and nearly every sector where information is required to be transferred. Images could also be interpreted as two-dimensional signals require analysing in many areas of technology which makes it important to develop processing methods that enhance their interpretibility in different forms.

Imaging techniques have come a long way ahead from its initial state [1]. However, noise generated in an image is due to technical limitations [2], unfavorable environment conditions, compression techniques and sometimes the choice of the photographer. Besides the last, it is favorable to remove the noise generated in those cases and hence many techniques have been developed over the years.

II. Analysis and Motivation

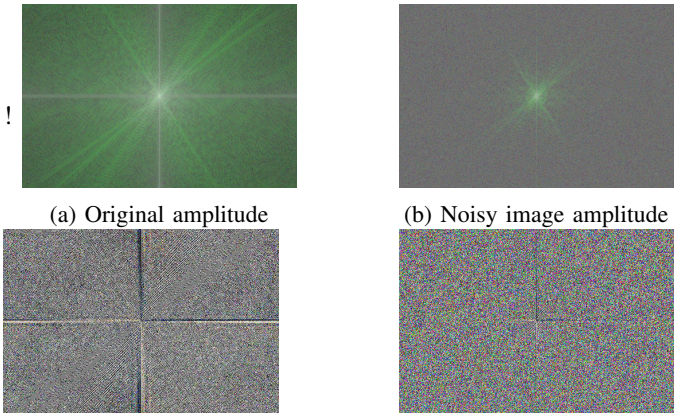
Data will be dealt with in grayscaled as well as the colored channels, we use the ITU-R 601-2 format $L = R * 0.299 + G * 0.587 + B * 0.114$ for color conversion into grayscale.

A. Noise pattern

We assume the model [3]

$$n(x, y) = g(x, y) - f(x, y) \tag{1}$$

$n(x, y)$ is defined as the noise defined either by a single value $\in \mathcal{R}$ or a 3-tuple $\in \mathcal{R}^3$ as is $g(x, y)$ which is the noisy image and $f(x, y)$ which is the original image. By doing this we establish that our data uses the **Gaussian noise** over all the images ($n(x, y) \sim \mathcal{N}(0, \sigma)$) in each of the RGB channels hence rendering extreme spikes. Since we are dealing in the Discrete domain, two-dimensional DFT becomes underwhelming as



(a) Original amplitude (b) Noisy image amplitude

(c) Original image phase (d) Noisy image phase

Fig. 1: Decay of features further from the center n-fft

$$G(u,v)=F(u,v)+\sum_{x=0}^{w}\sum_{y=0}^{h}\mathcal{N}(0,\sigma^2)exp\big(-2\pi i\big(\tfrac{ux}{w}+\tfrac{vy}{h}\big)\big) \tag{2}$$

where $G(u, v)$, $F(u, v)$ to be the 2d discrete fourier transformation of the noisy image and the original image respectively. The $\mathcal{N}(0, \sigma^2)$ value is affected by spatial dimensions, so it will generate different values for each of x,y (except for 0,0). Hence we are left with the option of only relying on applying filters on the fourier transform of the noisy images.

B. Omission of preprocessing techniques

Since we are dealing with RGB channels as well, any sort of preprocessing will lead to pixel differences that we will not be able to justify for our filter being chosen. These values are capped at their maximum values. Considering our noise model $n(x, y)$ will take on extra terms and hence it cannot be statisticaly analysized for an error-type. We also argue that for an initial analysis, our filter fares really well even without any of the preprocessing techniques on RGB channels.

C. nFFT on colored images

While dealing with multichannel case, choosing nFFT over FFT will considerably condense the image information into the center of the 3 dimensional form and applying filters and inverse transform on it will not cause visible **biased artifacting**. We have also observed that the images are usually on the RGB domain, the green comes out more prominent in

(a) nFFT and FFT (150,150,150) - Gives same image

(b) nFFT (100,150,100) - even artificating due to all channels

(c) FFT(100,150,100) - uneven artifacting due to green (middle) channels

Fig. 2: Applying cubic fourier filter of size in three channels

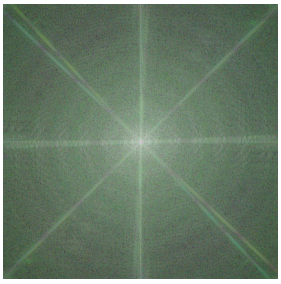
the n-FFT transform due to it being the middle layer (after the fourier shift).

With the increase in dimensionality one could think of extending any the 2d filter into their corresponding 3d form (a simple one would be for a circle/square in 2d fourier domain to be a sphere/cube/diamond in 3d).

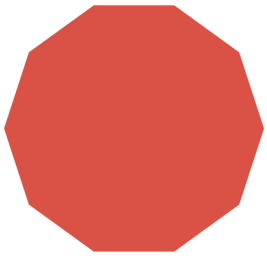
D. Features to be preserved

nFFT transform of the phase yields really weird With regular polygons one can immediately observe that the amplitude of their fourier transform will be highlighted by its edges which are inclined at the same angle for an image. With each

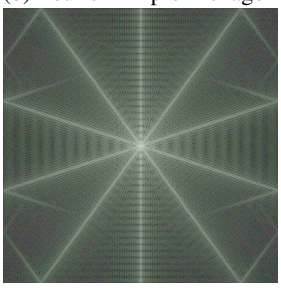
(a) Hexagon



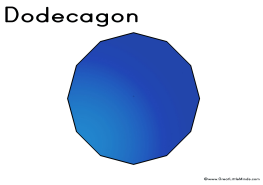
(b) Fourier Amp of hexagon



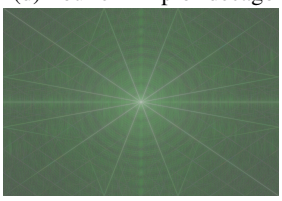
(c) Decagon



(d) Fourier Amp of decagon



(e) Dodecagon



(f) Fourier Amp Dodecagon

Fig. 3: Seeing patterns in the amplitudes of the fourier transform of even sided regular polygons

increasing even sided polygon, we can see that the angles it covers are evenly spread in its amplitude which leads us into considering a shape that rotates around the center point of the image. For reference, we will call this the "flower transform" as it somewhat looks like the petals of a flower rotated about a single axis.

III. Methodology

Our main analysis is done on google colab on a python-3.8 [**?**] environment with the skimage [4], numpy [5], PIL [6] libraries. Skimage was used only for metrics and comparison with reference generated images, PIL was used for better visualization and converting into grayscaled values and Numpy was used extensively for the fourier transform. All images used belong to the Berkeley Segmentation Dataset, we deal with a small subset of 25 images [7] .We will be creating 8 different evaluations based on the filters that we are using (Soft/Hard filter, Inclusion of phase removal and Median filter as a post processing step).

A. Creating the flower pattern

The choice of the petal of the flower is crucial as it dictates how much and where the amplitude taken would be the highest or least. This petal is then rotated about the center to create a generalized pattern. A problem arises when this petal's interior

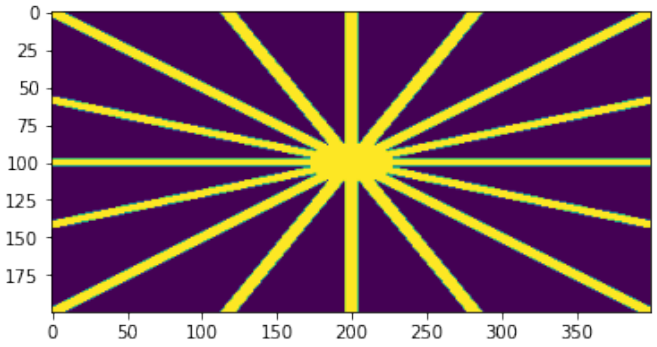


Fig. 4: **Flower transform of rectangle petal** - 2d filter considered

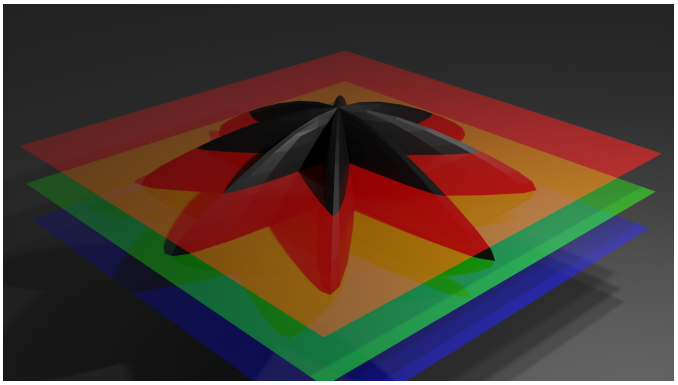


Fig. 5: **3 dimensional flower transform of rectangle petal** - 3d filter considered

area cannot be expressed explicitly. In that case, one needs to keep a copy of the original petal using hand-drawn methods into a two dimensional array. Affine transformations over the single petal of the flower helps us create the pattern that we require, however we need the single petal to generally consider all $k\pi/n$ angles where $n, k \in \mathcal{N}$. This leads to the number of petals we will be considering in the form of 2^k. For our case, we will be just using a simple rectangle whose area can be expressed explicitly.

Since there are fewer features further from the center we go, in cases of colored images we chose the fourier filter to scale down further we go from the center hence scaling it accordingly. The fourier filter hence generated in the 3-dimensional domain, is only one of the 3d transforms of the filter, there can be multiple forms of this.

B. Soft filter and Phase removal inclusion

Since the further we go from the center, there are higher amplitudes and these features are usually not required in our filter domain, we will consider a gaussian blur around the center of our filter with a sigma (= 7, chosen arbitarily). In cases of a hard filter (filter with only 0 and 1 values), the complete masked fourier image should remain the same as if only the amplitude goes to 0 then there is no point in considering $Ae^{i\phi}$ term as it goes to zero regardless of the phase

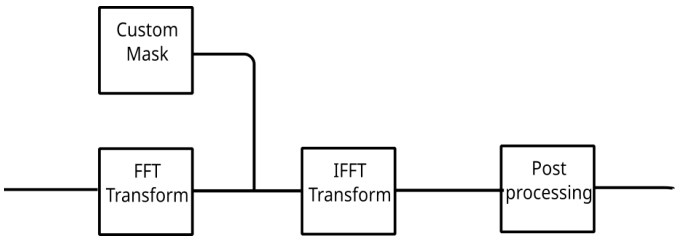


Fig. 6: **Flowchart** - Summary of the process

value. However in case of the soft filter with the amplitude taking cases in between 0 and 1, we see that inclusion of phase on the filter is reasonable as the values in between 0 and 1 in the filter, there can be different outputs.

C. Postprocessing

As there might remain some artifacting in the fourier transform of the images as shown previously we will consider using the median filter (square with width 3 in 2d and box with width 3 in 3d) as the last processing step. The choice of the median filter is simply due to artifacting spikes and the median filter being able to supress the noise in a single pass much better than the mean filter.

IV. Experiments

The filter chosen was generated keeping grayscale by caluculating 10 minimum PSNR values over width of the rectangular filter mentioned. We get values of length close (0.892) to the maximum dimension of the image and rectangle width of near 5. Hence we chose our basic filter. For the 3d transform, applying a simple grid search on the scale of the shrinkage of our chosen filter in the 0 and 2 index while keeping it the same in the 1 index. We also choose the sigma of our gaussian filter on the soft filter to be 7. We will be comparing our images against the mean, median, VisuShrink [8] and BayesShrink [9].

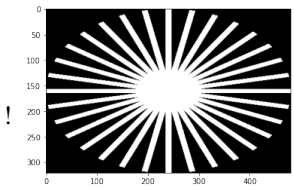
V. Conclusion

We outperform largely in case of colored over the BayesShrink images and have an almost similar in case of grayscaled as compared to BayesShrink. However, these metrics usually do not work in the domain of multichannel features. We find ourselves with a pretty nice filter that once generated works as fast as any other fourier filter and produces great results. For our purposes we have used only used a rectangular shaped petal, one could use an arbitary curve for the initial petals obtain even better results.

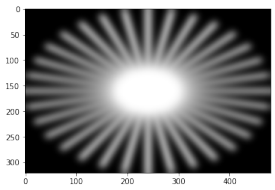
VI. Future directions

A. Faster generation of the filter

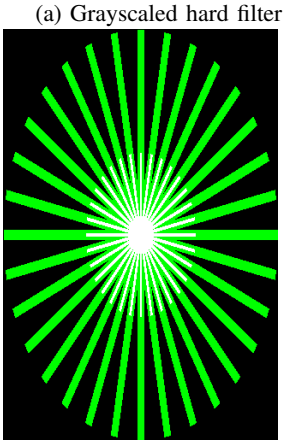
Since our filter uses affine transformation and per pixel checking, the time required for the generation of the filter is $O(n^2)$ which on the data takes about 3 minutes (with default python) to generate a single filter. This could be increased with precompiled code or interpolating the image after using some points.



(a) Grayscaled hard filter



(b) Grayscaled soft filter



(c) Colored hard filter

(d) Colored soft filter

Fig. 7: Final chosen filters

B. Better selection of the flower angles

We have used a general transformation that is symmetric about the origin, however with rotated pictures, we might need some form of initial detection of the orienation of the image and then act upon it. An even better selection of the filter would be for it to detect the angles where the prominent feature are automatically and then act upon it.

C. Complex summation and Gaussian noise validation

In case, we know that the noise type we are dealing with is indeed normal then using the squared values of the bivariate-normal distribution (Rice distribution [10]) one could possibly obtain the potential sigma values which could help direct the shape of the generalised filter. However with our data it was a realisation and the time frame fell short of creating this optimization route.

D. Voxel based denoising patterns

As our methods here have been generalised to the nFFT domain from a 2d filter. We could possible use the same blurring pattern into higher dimensions which after an image pixels is usually 3d scene voxel. There has been multiple research in denoising by large tech companies. Our approach seems fovorable due to the convolution in the 2d domain however, converting a 3d scene into its fourier form and back might be another overhead of $O(n^2 log(n))$.

(a) Tiger image

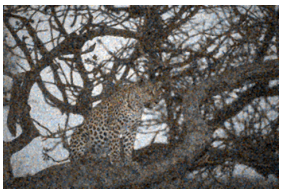
(b) Castle Image

(c) Noisy image

(d) Noisy Image

(e) Mean Filter

(f) Mean Filter



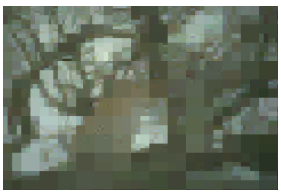
(g) Median Filter

(h) Median Filter

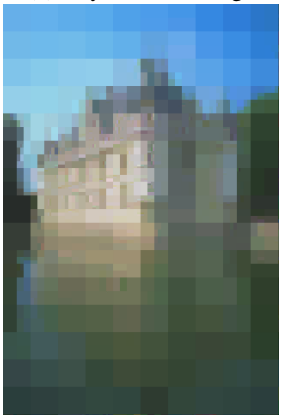
Fig. 8: Comparing results

(a) BayesShrink Image

(b) BayesShrink Image



(c) VisuShrink Image



(d) VisuShrink Image

(e) Our method (+ soft + only amplitude)

(f) Our method (+ soft + only amplitude)

(g) Our method (+ soft + only amplitude + median)

(h) Our method (+ soft + only amplitude + median)

Fig. 9: Comparing results - Our top 2 results according to algorithm

REFERENCES

[1] M. McFarlane, "Digital pictures fifty years ago," vol. 60, no. 7, pp. 768–770. Conference Name: Proceedings of the IEEE.

[2] A. C. Bovik, *Handbook of Image and Video Processing*. Academic Press. Google-Books-ID: OYFYt5C4N94C.

[3] N. M. Blachman, "On fourier series for gaussian noise," vol. 1, no. 1, pp. 56–63.

[4] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 6 2014.

[5] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[6] P. Umesh, "Image processing in python," *CSI Communications*, vol. 23, 2012.

[7] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, pp. 416–423, July 2001.

[8] D. L. Donoho and I. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, pp. 425–455, 09 1994.

[9] S. Chang, B. Yu, and M. Vetterli, "Adaptive wavelet thresholding for image denoising and compression," *IEEE Transactions on Image Processing*, vol. 9, no. 9, pp. 1532–1546, 2000.

[10] S. O. Rice, "Mathematical analysis of random noise," *The Bell System Technical Journal*, vol. 23, no. 3, pp. 282–332, 1944.

TABLE I: Mean values of SSIM over the grayscaled dataset and generated images

Dataset	noisy10	noisy25	noisy50
GS SSIM	0.7962	0.6303	0.4299
GS SSIM (+ Only amp)	0.7962	0.6303	0.4299
GS SSIM (+ Median)	0.8098	0.7302	0.5817
GS SSIM (+ Only amp + Median)	0.8098	0.7302	0.5817
GS SSIM (+ Blur)	0.8160	0.7143	0.5434
GS SSIM (+ Blur + Only amp)	0.8943	0.7534	0.5487
GS SSIM (+ Blur + Median)	0.8101	0.7480	0.6182
GS SSIM (+ Blur + Only amp + Median)	0.8448	0.7711	0.6276
Mean filter SSIM	0.8442	0.7535	0.5893
Median filter SSIM	0.8460	0.7171	0.5241
BayesShrink SSIM	0.9089	0.7759	0.6409
VisuShrink SSIM	0.6810	0.5566	0.4709
Original SSIM	0.83176	0.5514	0.3265

TABLE II: Mean values of MSE over the grayscaled dataset and generated images

Dataset	noisy10	noisy25	noisy50
GS MSE	148.69	228.20	498.87
GS MSE (+ Only amp)	148.69	228.20	498.87
GS MSE (+ Median)	141.61	170.07	274.35
GS MSE (+ Only amp + Median)	141.61	170.07	274.35
GS MSE (+ Blur)	140.70	170.36	304.63
GS MSE (+ Blur + Only amp)	67.98	116.23	281.24
GS MSE (+ Blur + Median)	150.33	172.15	259.00
GS MSE (+ Blur + Only amp + Median)	120.62	147.83	240.24
Mean filter MSE	125.15	155.65	264.89
Median filter MSE	104.84	152.18	300.71
BayesShrink MSE	33.02	114.39	263.35
VisuShrink MSE	218.38	388.67	610.52
Original MSE	44.04	263.35	953.55

TABLE III: Mean values of SSIM over the colored dataset and generated images

Dataset	noisy10	noisy25	noisy50
COL SSIM	0.7887	0.6361	0.4424
COL SSIM (+ Only amp)	0.7887	0.6361	0.4424
COL SSIM (+ Median)	0.8167	0.6800	0.4929
COL SSIM (+ Only amp + Median)	0.8167	0.6800	0.4929
COL SSIM (+ Blur)	0.8077	0.7215	0.5652
COL SSIM (+ Blur + Only amp)	0.8934	0.7672	0.5724
COL SSIM (+ Blur + Median)	0.7880	0.7396	0.6309
COL SSIM (+ Blur + Only amp + Median)	0.8219	0.7631	0.6414
Mean filter SSIM	0.8240	0.7367	0.5791
Median filter SSIM	0.8167	0.6800	0.4929
BayesShrink SSIM	0.8580	0.6977	0.5633
VisuShrink SSIM	0.6246	0.5045	0.4361
Original SSIM	0.7213	0.4130	0.2167

TABLE IV: Mean values of MSE over the dataset and generated images

Dataset	noisy10	noisy25	noisy50
COL MSE	200.51	264.01	513.69
COL MSE (+ Only amp)	200.51	264.01	513.69
COL MSE (+ Median)	151.29	217.95	395.29
COL MSE (+ Only amp + Median)	151.29	217.95	395.29
COL MSE (+ Blur)	172.40	185.03	304.03
COL MSE (+ Blur + Only amp)	77.886	120.08	301.61
COL MSE (+ Blur + Median)	199.90	218.40	282.58
COL MSE (+ Blur + Only amp + Median)	168.57	193.11	395.29
Mean filter MSE	270.84	302.42	416.67
Median filter MSE	151.29	217.95	395.29
BayesShrink MSE	57.98	178.77	360.72
VisuShrink MSE	279.53	491.75	736.98
Original MSE	97.280	580.35	2067.19