6.300: Signal Processing

Filtering and Inverse Filtering

November 24, 2026

Image Processing

Example: Remove high frequencies from an image.



The most straightforward approach is to take the Fourier transform, zero out the high-frequency components, and then inverse transform.

Image Processing

Transform, zero out the high-frequency components, and inverse transform.

```
from lib6300.fft import fft2,ifft2
from lib6300.image import png_read,show_image

f = png_read('bluegill.png')
R,C = f.shape

F = fft2(f)                              # take 2D DFT
for kr in range(-R//2,(R+1)//2):         # loop over frequencies
    for kc in range(-C//2,(C+1)//2):
        if (kr**2+kc**2)**0.5 > 25:      #   if this is a high frequency
            F[kr,kc] = 0                 #      zero its amplitude
filtered = ifft2(F)
show_image(filtered)
```

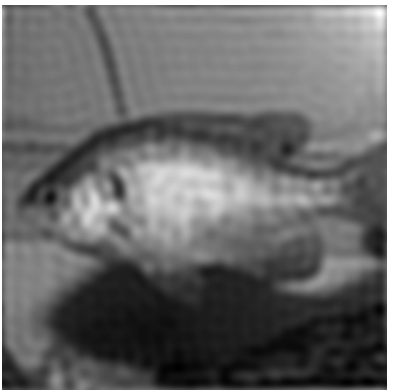
Image Processing

Transform, zero out the high-frequency components, and inverse transform.

original



filtered



Why are there ripples in the "filtered" version?

The ripples look like waves. How could ripples arise?

Image Processing: 2D Filtering

Zeroing out frequency components is equivalent to filtering.

$$H_L[k_r, k_c] = \begin{cases} 1 & \text{if } \sqrt{k_r^2 + k_c^2} \leq 25 \\ 0 & \text{otherwise} \end{cases}$$

```
f = png_read('bluegill.png')
R,C = f.shape
F = fft2(f)

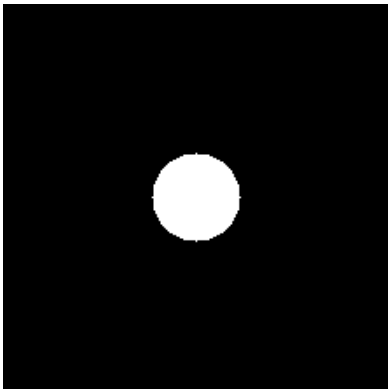
HL = numpy.zeros((R,C),dtype=complex)
for kr in range(-R//2,(R+1)//2):
    for kc in range(-C//2,(C+1)//2):
        if (kr**2+kc**2)**0.5 <= 25:
            HL[kr,kc] = 1

lowpassed = ifft2(F*HL)
show_image(lowpassed)
```

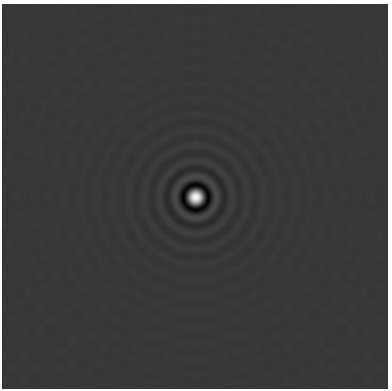
2D Filtering

Compare frequency view ($H_L[k_r, k_c]$ to spatial view ($h_L[r, c]$). Filtering is multiplication in the frequency domain, or equivalently convolution in space.

Frequency View



Spatial View



Step changes in $\left|H_L[k_r, k_c]\right|$ $\rightarrow$ overshoot in $h_L[r, c]$: Gibb's phenomenon.

2D Filtering

To reduce the ripples in space, we must limit the step discontinuities in the transform. Try a Hann window.

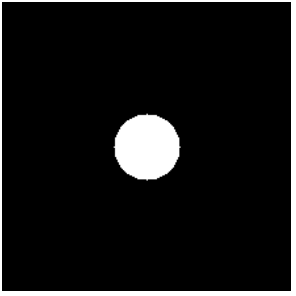
$$H_{L2}[k_r, k_c] = \begin{cases} \frac{1}{2} + \frac{1}{2}\cos\left(\pi \times \frac{\sqrt{k_r^2+k_c^2}}{50}\right) & \text{if } \sqrt{k_r^2+k_c^2} \leq 50 \\ 0 & \text{otherwise} \end{cases}$$

```
HL2 = numpy.zeros((R,C),dtype=complex)
for kr in range(-R//2,(R+1)//2):
    for kc in range(-C//2,(C+1)//2):
        d = (kr**2+kc**2)**0.5
        if d<=50:
            HL2[kr,kc] = 0.5+0.5*cos(pi*d/50)

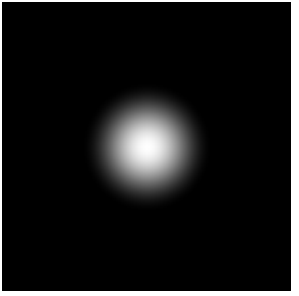
hanned = ifft2(F*HL2)
show_image(hanned)
```

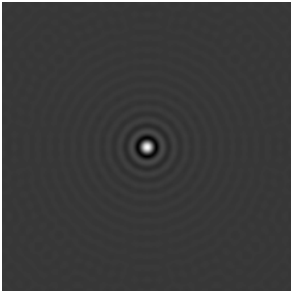
Comparing Filters

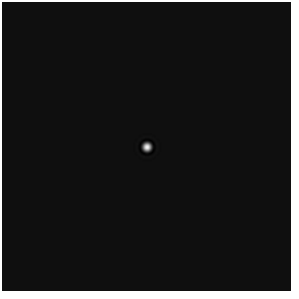
Square Window



Hann Window



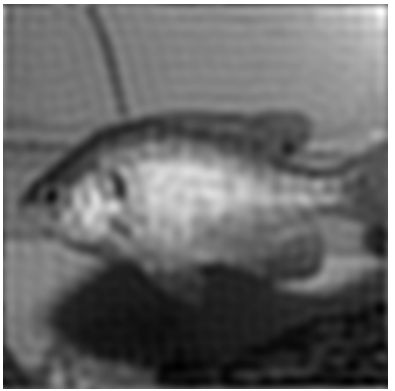




2D Filtering

Ripples are gone from the Hann window.

Square Window



Hann Window

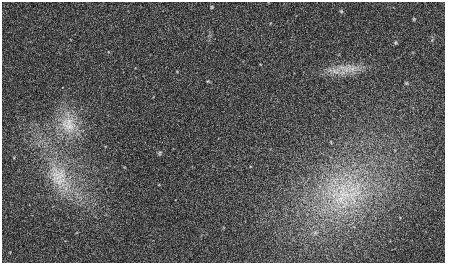


But the the image is still blurry!

Is lowpass filtering useful?

Reducing Noise

Effect of low-pass filtering of telescope image of night-time sky.





Removing Artifacts

Original high-resolution (5880x3920) image of Zakim Bridge.



Removing Artifacts

Why does downsampling generate artifacts?

(1470x980)



(735x490)



(368x245)



(184x123)



Down-Sampling Artifacts

Down-sampling artifacts in images and music: similar mechanisms?

Image artifacts:



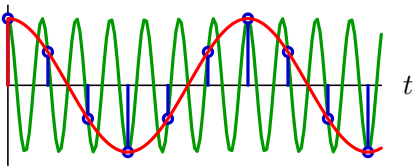
Music artifacts:

- $f_s = 44.1$ kHz
- $f_s = 22$ kHz
- $f_s = 11$ kHz
- $f_s = 5.5$ kHz
- $f_s = 2.8$ kHz

J.S. Bach, Sonata No. 1 in G minor Mvmt. IV. Presto
Nathan Milstein, violin

Sampling Artifacts

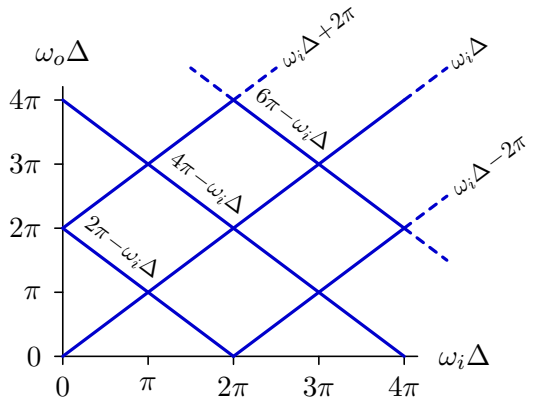
Aliasing occurs when samples (shown in blue) of two different frequencies (green and red) are identical.



Each of these CT frequencies is an **alias** of the other.

Sampling Artifacts

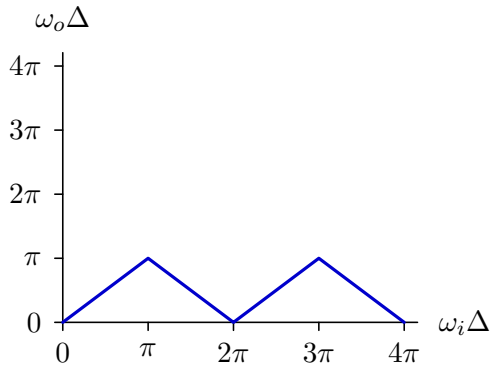
Sampling introduces new frequencies ω_o not in the signal being sampled.



Each point on each of these blue lines represents a pair of frequencies ω_i and ω_o that produce the same discrete-time samples when the sampling interval is Δ.

Sampling Artifacts

Frequencies $\omega_o > \pi/\Delta$ can be removed by low-pass filtering after sampling.

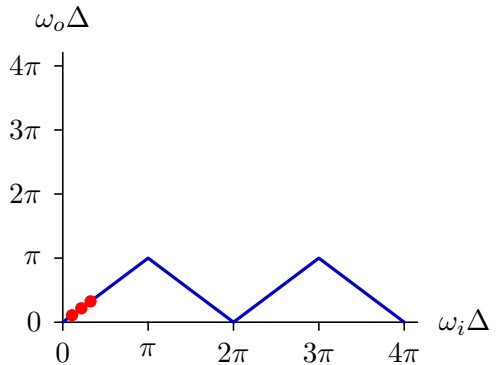


$\omega_o > \pi/\Delta = \pi f_s$

$f_o > f_s/2$

Sampling Artifacts

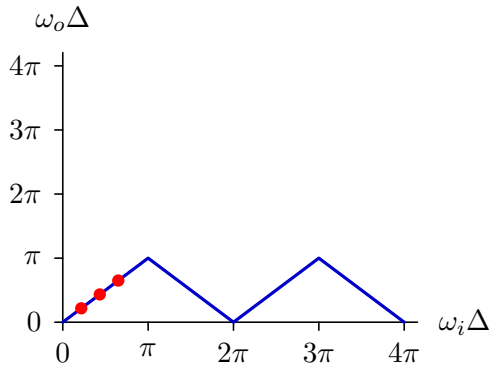
Even if we restrict the output to lie in the base band, harmonically related input frequencies may give rise to inharmonic output frequencies.



Here, three harmonically related input frequencies at low frequencies give rise to three harmonically related output frequencies.

Sampling Artifacts

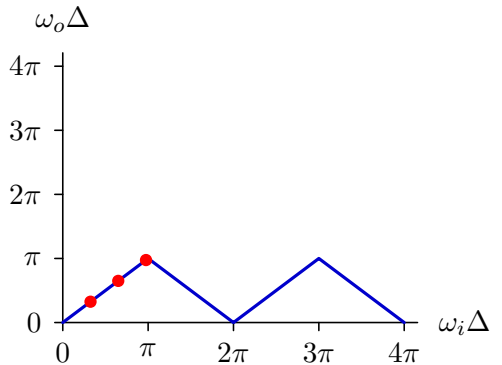
Even if we restrict the output to lie in the base band, harmonically related input frequencies may give rise to inharmonic output frequencies.



Higher input frequencies also produce harmonically related output frequencies.

Sampling Artifacts

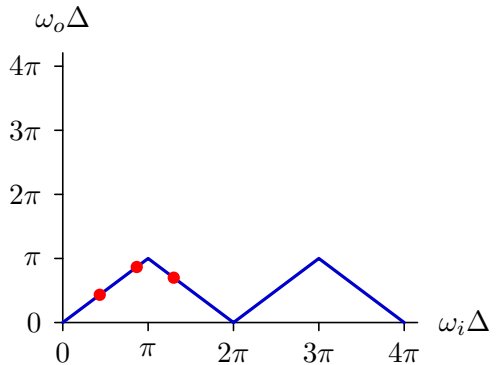
Even if we restrict the output to lie in the base band, harmonically related input frequencies may give rise to inharmonic output frequencies.



Higher input frequencies also produce harmonically related output frequencies.

Sampling Artifacts

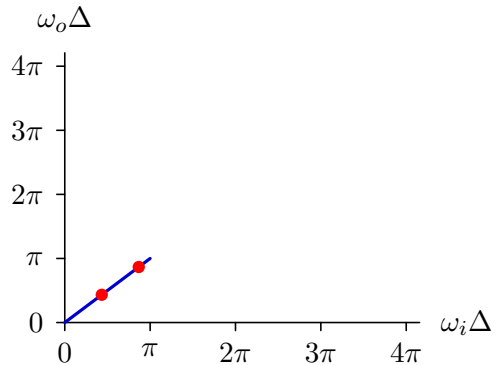
Even if we restrict the output to lie in the base band, harmonically related input frequencies may give rise to inharmonic output frequencies.



Still higher input frequencies produce output frequencies that are not found in the input.

Sampling Artifacts

Anti-aliasing refers to lowpass filtering the input **before** sampling to remove components that would alias to frequencies $\omega_o \neq \omega_i$.



Still higher input frequencies produce output frequencies that are not found in the input.

Anti-Aliasing

Sampling artifacts in music reduced by anti-aliasing (LPF before sampling).

- $f_s = 11$ kHz without anti-aliasing
- $f_s = 11$ kHz with anti-aliasing
- $f_s = 5.5$ kHz without anti-aliasing
- $f_s = 5.5$ kHz with anti-aliasing
- $f_s = 2.8$ kHz without anti-aliasing
- $f_s = 2.8$ kHz with anti-aliasing

J.S. Bach, Sonata No. 1 in G minor Mvmt. IV. Presto
Nathan Milstein, violin

Down-Sampling and Aliasing

Can similar anti-aliasing techniques work for images?

(1470x980)



(735x490)



(368x245)



(184x123)



Aliasing in Images

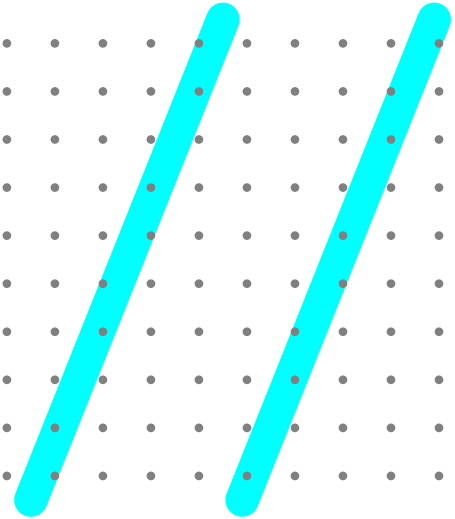
While the results look different, the mechanisms underlying aliasing in music and images are similar.



Start with an image with two (cyan) lines

Aliasing in Images

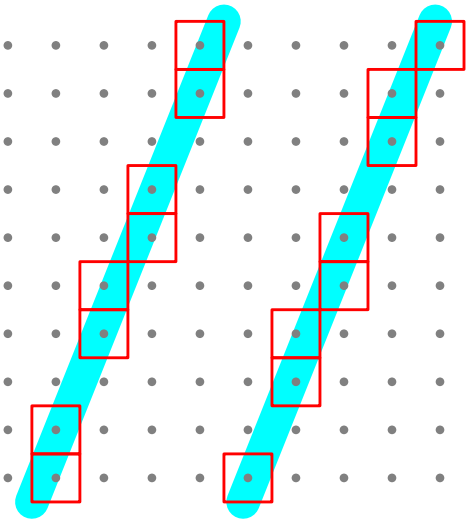
While the results look different, the mechanisms underlying aliasing in music and images are similar.



Now sample uniformly with sampling interval Δ horizontally and vertically.

Aliasing in Images

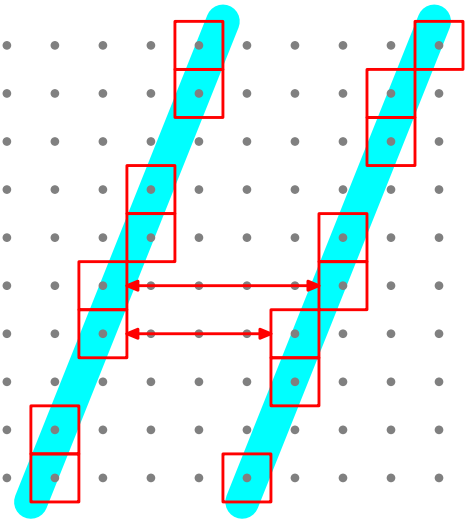
While the results look different, the mechanisms underlying aliasing in music and images are similar.



The red boxes indicate pixels whose centers fall on the cyan lines.

Aliasing in Images

While the results look different, the mechanisms underlying aliasing in music and images are similar.



Aliasing affects periodicity (arrows) in music and in images.
But aliasing drastically affects edges in images.

Will lowpass filtering before sampling help? Why? Why not?

Down-Sampling and Aliasing

Down-sampling after low-pass filtering (Hann window, half-width = 150).

(1470x980)



(735x490)



(368x245)



(184x123)



High-Pass Filtering

Use the same approach to implement a high-pass filter.

$$H_H[k_r, k_c] = 1 - H_L[k_r, k_c] = \begin{cases} 1 & \text{if } \sqrt{k_r^2 + k_c^2} > 25 \\ 0 & \text{otherwise} \end{cases}$$

In the spatial domain, then, we have:

$$h_H[r, c] = RC\delta[r, c] - h_L[r, c]$$

High-Pass Filtering

Not surprisingly, results show the same rippling effect seen in LPF.



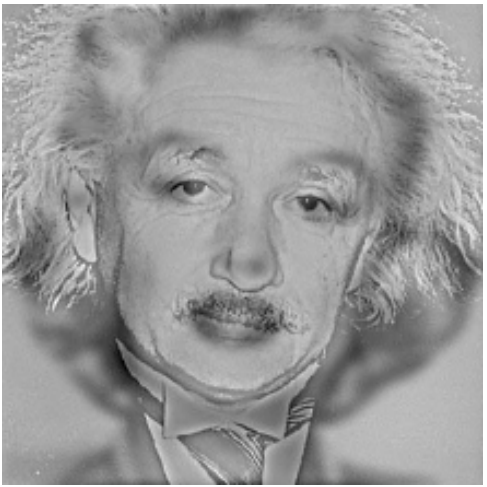
High-Pass Filtering

We can reduce the ringing artifacts by using $1 - H_{L2}[k_r, k_c]$ instead.



Who Is This?

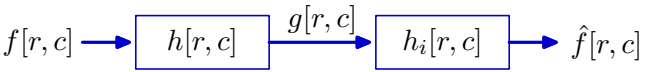
Look at this image with your eyes about a foot away from the screen. Then look again from a distance of six feet.



from Prof. Antonio Torralba

Filtering and Inverse Filtering

An important area of research in image processing is in **inverse filtering** (also called deconvolution). The idea is to undo the effect of prior filtering.



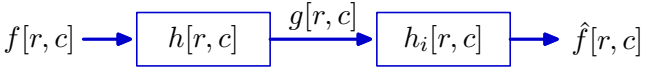
Example: enhancing images from a telescope.

- $f[r,c]$: unknown image of a distant galaxy and
- $h[r,c]$: effects of optics of the telescope (especially lowpass filtering)

Goal: design an inverse filter $h_i[r,c]$ so that $\hat{f}[r,c]$ approximates $f[r,c]$.

Inverse Filtering

One simple approach is to filter by the inverse of $H[k_r, k_c]$.



In the frequency domain:

$$\hat{F}[k_r, k_c] = H_i[k_r, k_c] \times G[k_r, k_c] = H_i[k_r, k_c] \times (H[k_r, k_c] \times F[k_r, k_c])$$

If $H_i[k_r, k_c] \times H[k_r, k_c] = 1$ then $\hat{F}[k_r, k_c] = F[k_r, k_c]$!

Letting $H_i[k_r, k_c] = \dfrac{1}{H[k_r, k_c]}$ is called **inverse filtering**.

Quite remarkable that you can design a system to undo the effect of a prior system. Think about how you might do "inverse convolution"!

But it's simple (?) in the frequency domain.

Example: Motion Blur

Camera images are blurred by motion of the target.

The resulting **motion blur** can be modelled as the convolution.



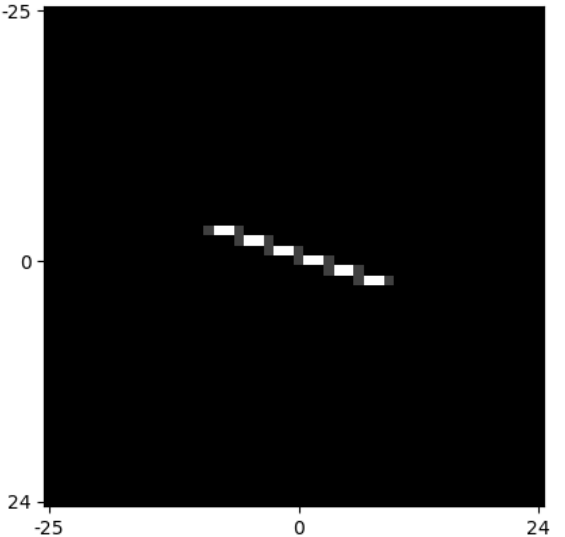
Modelling Motion Blur

Assume that streaks in this image resulted from the blurring. There is an isolated streak near the point r=120, c=250 (approximate 19x6 pixels).



Inverse Filtering

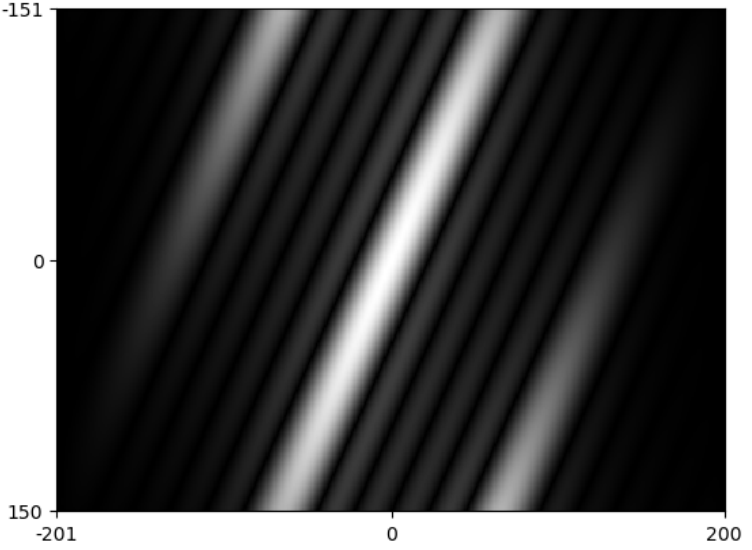
Make an image h to represent the presumed blurring function.



Let H represent the DFT of h, and filter the blurred image with $\frac{1}{H}$.

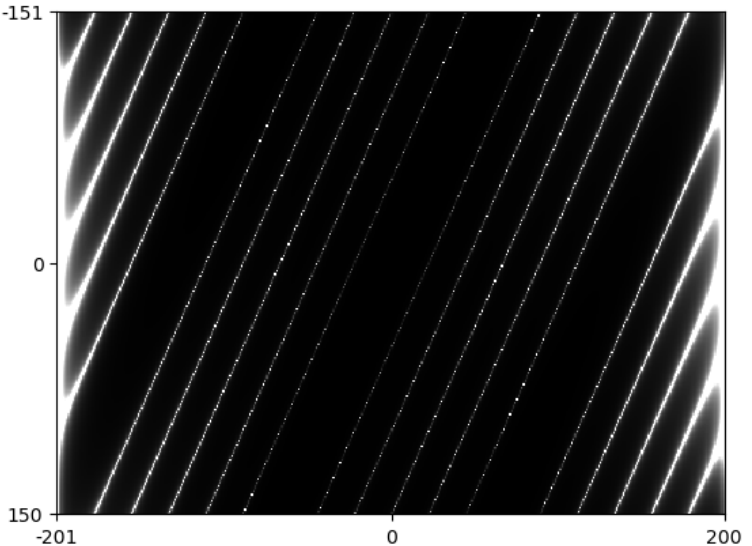
Inverse Filtering

This image shows the magnitude of H (DFT of blur function).



Inverse Filtering

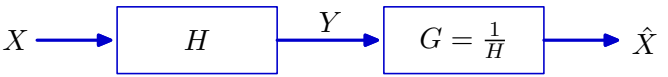
This image shows the magnitude of $\frac{1}{H}$.



What causes the bright spots? Why are they a problem?

Deblurring

The bright spots in $\frac{1}{H}$ come from points in H with values near zero.



Such bright spots dominate the result. Try limiting their magnitudes.

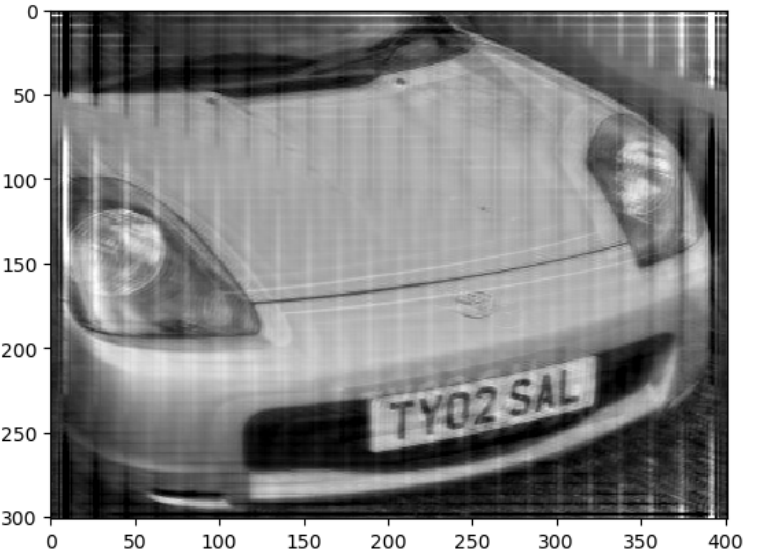
Method 1:

Start with $G = \dfrac{1}{H}$, but limit the magnitude of every point in G to 4:

```
for kr in range(R):
    for kc in range(C):
        G[kr,kc] = 1/H[kr,kc]
        if abs(G[kr,kc])>4:
            G[kr,kc] *= 4/abs(G[kr,kc])
```

Deblurring

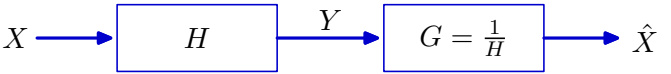
This deblurring filter works better: easy to read license number.



But there are many artifacts.

Deblurring

The form of the previous deblurring function is a bit arbitrary.



Method 2:

Here is a frequently used alternative (a "Weiner filter"):

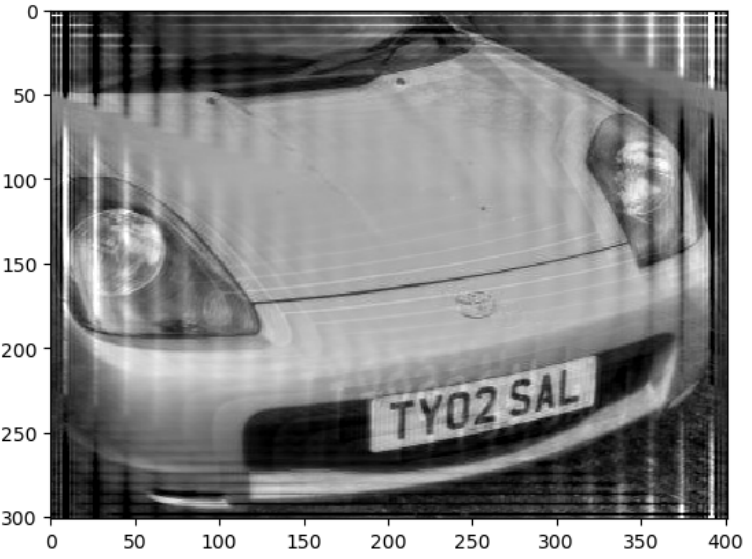
$$G = \frac{1}{H}\frac{|H|^2}{|H|^2 + C}$$

where $C = 0.004$ (chosen by trial and error).

Deblurring

Alternative deblurring function.

But there are still artifacts.



Edge Effects

Much of the ringing results from circular convolution.



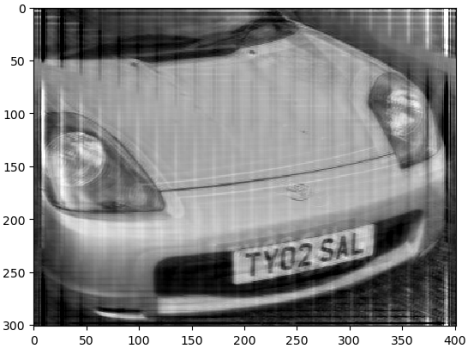
Edge Effects

Much of the ringing results from circular convolution. Window edges in original image to reduce step change due to periodic extension.



Comparison

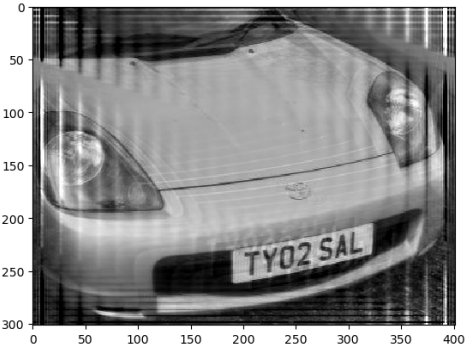
Method 1 with and without windowing.

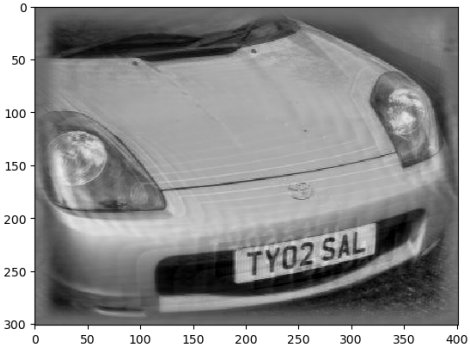




Comparison

Method 2 with and without windowing.





Conclusions

In general, inverse filtering worked well. It allowed a clear view of the license plate which was otherwise not legible.

Problems with inverse filtering. Inverting $H[k_r, k_c]$ doesn't work well if $H[k_r, k_c]$ is near zero. Fortunately, there were only a few such points. Arbitrarily limiting the values of such points results in useful deblurring.

Problems with circular convolution. Circular convolution introduces enormous artifacts if the left and right (or top and bottom) edges differ in brightness. These artifacts can be reduced by windowing.

Remaining problems. The resulting images still suffer from ringing – presumeably because of sharp discontinuities in the frequency representation of blurring.