Digital Image Processing

(1)

Lecture 8

Fourier world of spatial filters

Frequency Domain Filtering

$$I(x,y) \rightarrow \boxed{\text{2D DFT}} \rightarrow I'(u,v) \rightarrow \boxed{\text{Filter } H(u,v)} \rightarrow G(u,v)$$

Input Image

$G(u,v) = I'(u,v)\,H(u,v)$

$$G(u,v) \rightarrow \boxed{\text{2D IDFT}} \rightarrow g(x,y)$$

OUTPUT Image

Spatial Domain Convolution $\Longleftrightarrow$ Freq Domain Multiplication

— Valley Image 4:24

```
Imshow (fftshift(Log(abs(fft2(im)))), [])
```

6:53

② Don't be fooled by the log scale that all of this stuff is happening at high frequency - in reality it's very small.

→ imshow(fftshift(out), [0, 500000])

- how wide is the light spot area

Sinput (2)

→ about 20 ~~pixels~~ units wide in freq domain

→ out = idealfilt(im, [20 20], 'lp');

frequency filter mask, pass all frequencies Through this box and stop everything else.

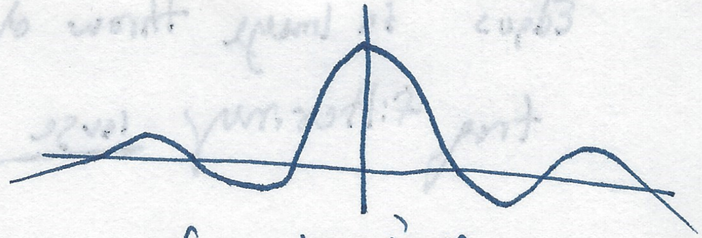
→ Take inverse DFT we have a blurry image 9:55

Loose high-freq (edge) stuff) (Ripple lines)

→ Spatial filter looks like in Spatial domain 10:11

Not a ~~small~~ smooth roll off

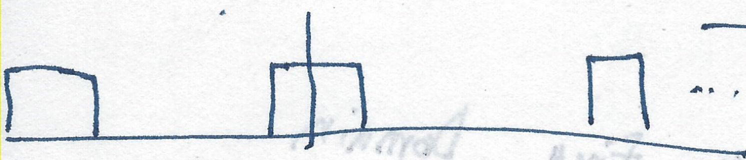
→



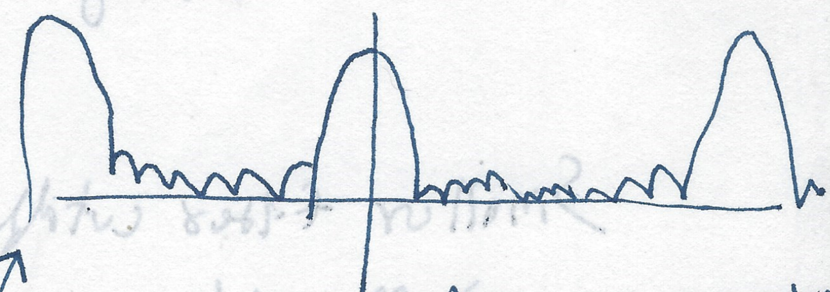
SPATIAL Domain Pulse

freq domain

Sinc



→



Discrete Pulse Train

Kind of like a periodic

Sinc

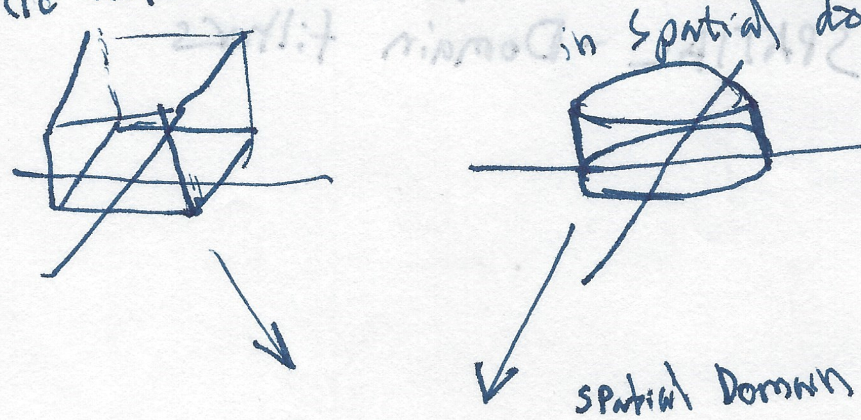
$\frac{\sin(\ )}{\text{sinc}(\ )}$

→ Spatial filter is Ripply like this

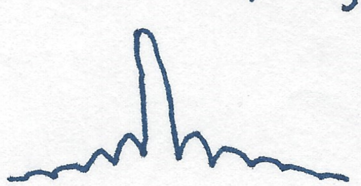
→ Big filter, Smoother spatial filter

→ Small filter, more $\frac{\sin}{\sin}$

→ circle filter → Radial symmetric Ripples in Spatial domain



spatial Domain filter,



⟹ "Ringing"

(4) Edges in image throw of these Bumps 19:45

freq filtering cause Ringing in images with a hard edge (Spatial domain)

To remove Ringing → (*)

Smaller filter cutoff in Freq Domain
⇒ More blurring
Bigger / More obvious Ringing

Period of Ringing proportional ~~#~~ to filter size. Smaller Period, bigger filter.

(*) ∴ To avoid ~~#~~ ringing, we often prefer Simple SPATIAL - Domain filters

T 23:31

~~X~~ Box in Spatial Domain → Rippleness in the frequency domain

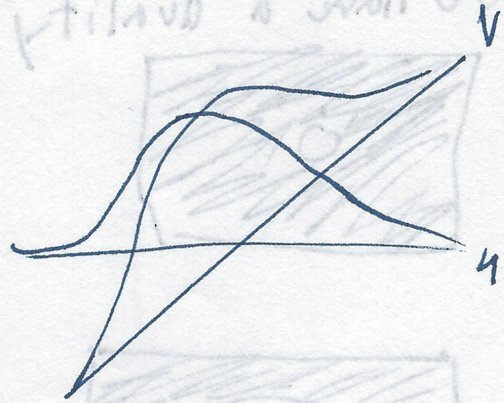
Ringing avoided via Filtering in the Spacial domain.

having a box filter in 1D filtering caused ringing Audio (In the frequency domain, box filter) in frequency domain

best of both worlds, ~~Good~~ exception,

→ Gaussian Low Pass filter

$$H(u,v) = e^{-\frac{u^2+v^2}{2\sigma^2}}$$



Gaussian in spatial Domain

↕

Gaussian in frequency domain

Gaussian filtered Image 27:45

~~increasing~~ $\cancel{\sigma^2}$,

having a smaller radius, smaller σ^2

we have a bigger domain in frequency world

→ So we let more information through

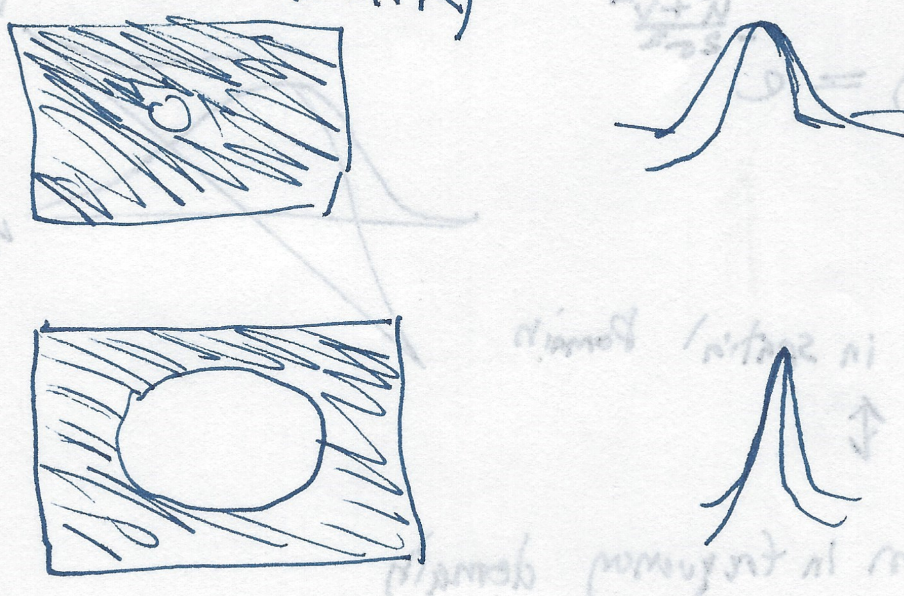
---

having a larger radius, σ^2, in the spatial domain

we have a smaller frequency domain

→ very blurry image with no ringing

* We have a duality



frequency

Spatial

doc fspecial

h= fspecial (type)

h= fspecial (type, Parameters)

Gaussian:

h= fspecial('gaussian', hsize, sigma)

h= fspecial('gaussian', [11 11], 2);

surf(h)

high Pass filtering

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \text{LowPass}$$

here's the Idea

LowPass + highpass = identity (whole thing)

as big as you want

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\,\text{ones}(3)$$

$$= \begin{bmatrix} -\frac{1}{9} & -\frac{1}{9} & -\frac{1}{9} \\ -\frac{1}{9} & \frac{8}{9} & -\frac{1}{9} \\ -\frac{1}{9} & -\frac{1}{9} & -\frac{1}{9} \end{bmatrix}$$

8 denote it as h → Look at 35:33

filter, Dark when small, white where large

DC value is zero, keeps values around peripheral

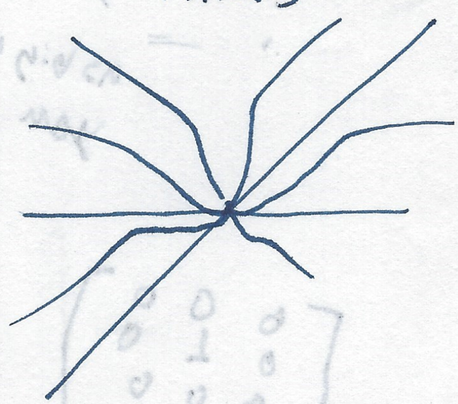
Different LPF

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} = Lp \rightarrow \begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix} - Lp = HP$$

? are the 3D models spatial Filters?

Now say

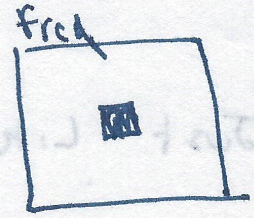
$$\begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix} - \text{LP Gaussian}$$



⇒ <u>inverted</u> Gaussian

<u>39:00</u> We note That we want to See the effects of the <u>HPF</u>

Say we have a box (in the freq domain where, the DC component is knocked out



So then we have just the edges

we have a ringing effect

```
h=fspecial('gaussian', [11 11], 2); % sums up to 1
id11=zeros(11); id11(6,6)=1;
hp11=id11-h;
out=imfilter(im,hp11);
```

Laplacian (when we got to filtering we mentioned this)

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

$$F(\nabla^2 I) = (2\pi)^2 (ju)^2 F(I) + (2\pi)^2 (jv)^2 F(I)$$

$$= \boxed{-4\pi^2(u^2+v^2)} F(I)$$

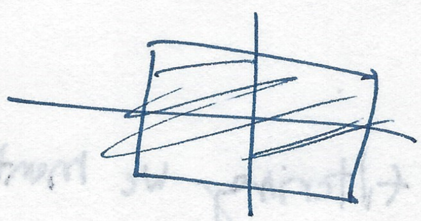
Zero in the middle

$H(u,v)$ - frequency domain version of Laplacian Filter h= fspecial ('laplacian')

Just like in 1D, we have to deal with Sampling, Aliasing, Band limited, Nyquist

e.g A 2D function (IMAGE) is <u>Band Limited</u> if ITS Fourier Transform is 0 outside some rectangular region, i.e.

$$F(u,v) = 0 \quad \text{for} \quad |u| \geq U_{max}, \quad |v| \geq V_{max}$$

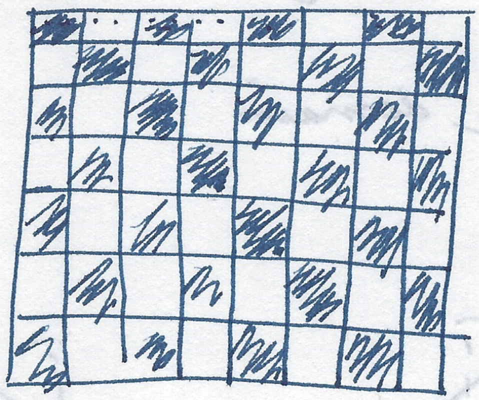


contain FT of a signal inside the box

2D Nyquist theorem

Sample Image ≥ to Nyquist frequency in both directions

we won't observe aliasing in resulting Image or Spatial aliasing.



Pic of black and white continuous time checkerboard

Sample higher than the highest frequency details in the Image

I suppose we can't help it if we lost some details.

ANTI-Aliasing

we don't take the aquired Image and then make it better

* Once an Image has aliasing in it, we can't undo it.

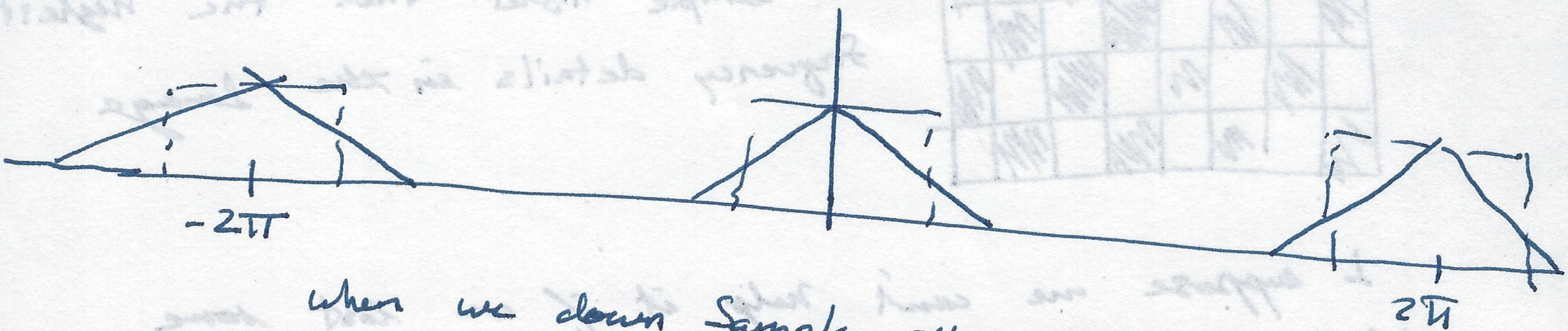
* ITS NOT REMOVING ALIASING from an Existing Image,

* ITS blurring a Digital IMAGE PRIOR To ReSAMPling to Avoid visual ARTIFACTS

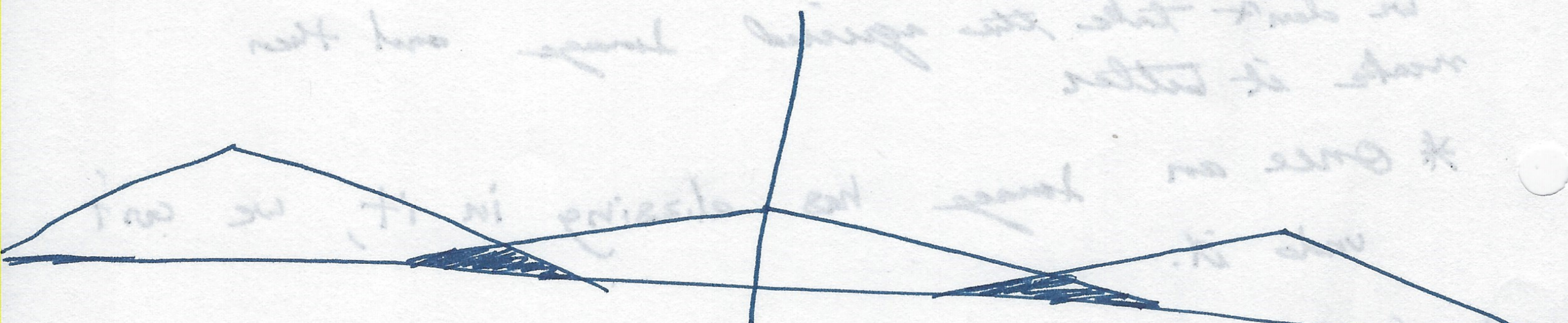
original IMAGE → $\boxed{\text{BLUR LPF}}$ → Down Sample

12 Recall from DSP UPSAMPLING and Down Sampling

Down Sampling in the frequency domain

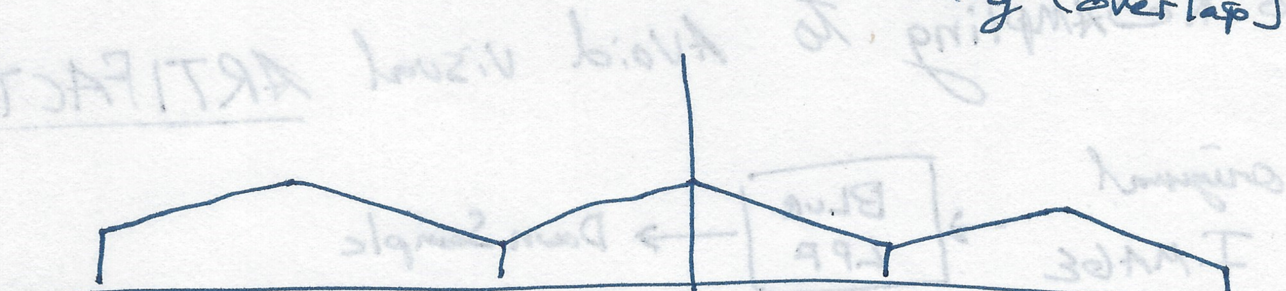


when we down sample, the copies shrink and spread out. (we can get aliasing because of this)



(we can see low frequency stuff turn into high frequency stuff.)

We apply a LPF prior to down sampling so that when we down sample, we have no aliasing (overlap)



Famous test Image with a woman

55:00

Take every 4th pixel of Image
(Down sampling by a factor of 4 in every direction)

out = im(1:4:end, 1:4:end);

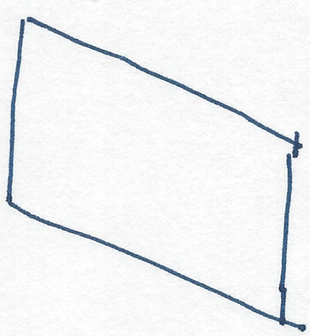
→ out put 55:40 Aliasing (we don't see vertical stripes anymore)

Solution
blur Image, then down Sample

→ We are going to lose the stripes anyway. So lose them before downsampling. (A trade off)

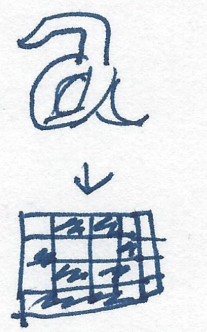
Anti-Aliasing IN VIDEO GAMES 58:10

Anti-Aliasing has many flavours (Also for Text)



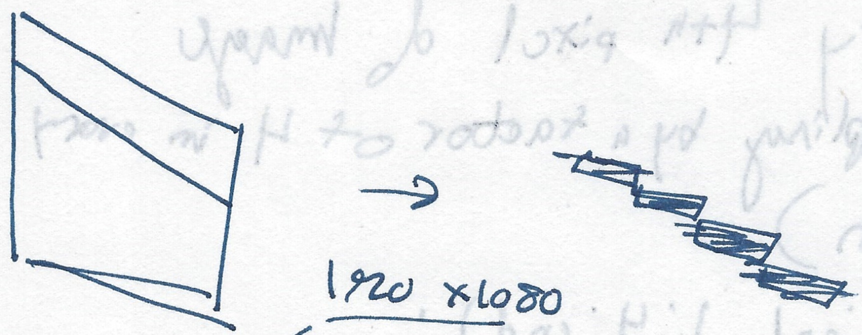
1920 x 1080

nice font (Sample coarsly)



from black and white to black and white
→ we really want Grey

we may go from a nice line to jagged edges



1920 x1080

render at $(1920 \times 4) \times (1080 \times 4)$ massive Image

↓ Down Sample each 4x4 square (e.g. Average) → blurring → LPF

To get each Low-Res Pixel

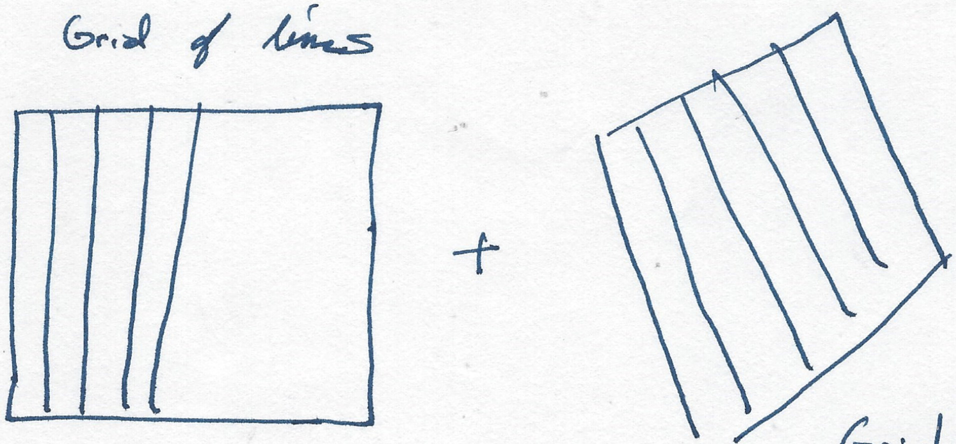
Investigating and Implementing different Kinds of down sampling normally occurring in graphics called Hard ware

→ try doing it in Matlab

Not a bad Idea

Moiri Patterns 1:03:

Grid of lines



Grid Rotated by θ°

As we rotate, the frequency shouldn't change

could mitigate this issue by LPF prior to rendering these images together.