Transform Methods & Signal Processing

lecture 06

Matthew Roughan
<matthew.roughan@adelaide.edu.au>

Discipline of Applied Mathematics
School of Mathematical Sciences
University of Adelaide

July 27, 2009

Block Diagrams

Block diagrams provide a visual metaphor for filters that can be useful in the design and analysis of filters. We can put together a more complex filter from simpler components as it we were putting together Lego.

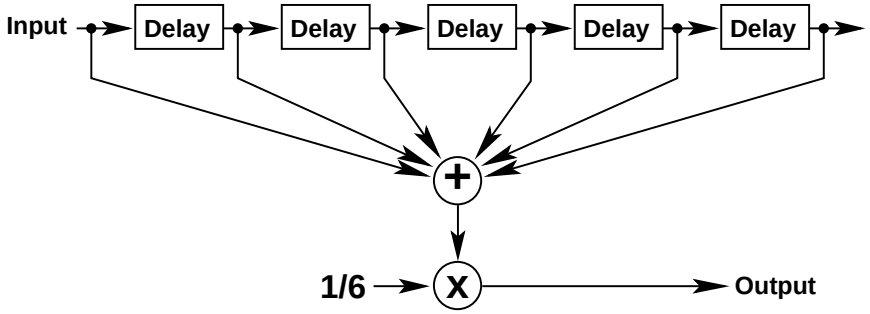
This lecture introduces block diagrams for representing and building filters. It also considers filtering in 2D and one of its applications: tomography, where we also see the Radon transform.

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.1/64

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.2/64

How to draw filters

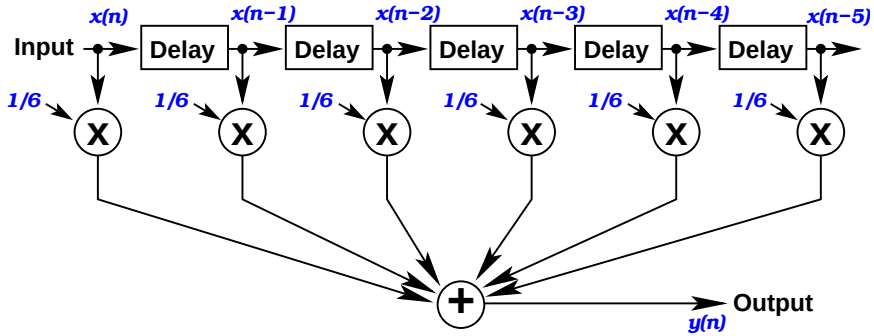
Block diagrams: example $y(n) = \frac{1}{6}\sum_{i=0}^{5} x(n-i)$



A six tap filter.

How to draw filters

Block diagrams: example $y(n) = \sum_{i=0}^{5} \frac{1}{6}x(n-i)$

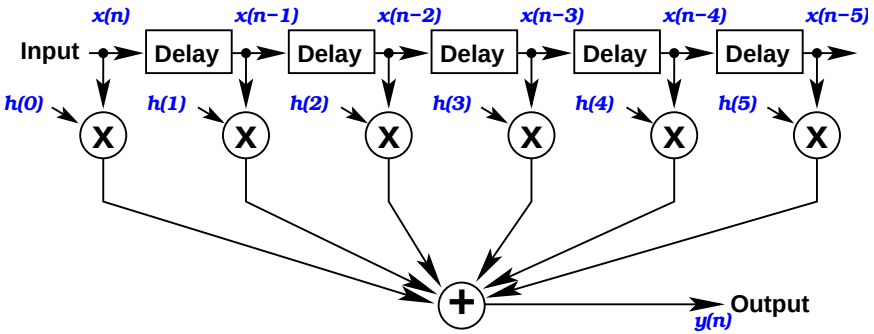


Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.3/64

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.4/64

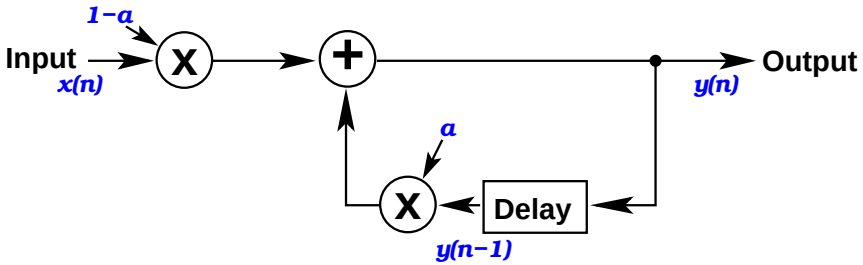
Block diagrams of FIR filters

FIR filter: $y(n) = \sum_{i=0}^{5} h(i)x(n-i)$



Block diagrams of IIR filters

IIR filter: $y(n) = ay(n-1) + (1-a)x(n)$



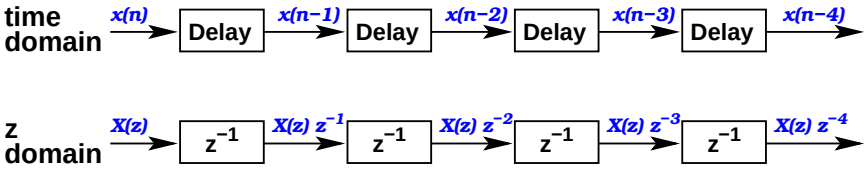
IIR filter designs use feedback (or recursion)

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.5/64

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.6/64

Block diagrams and Z-transforms

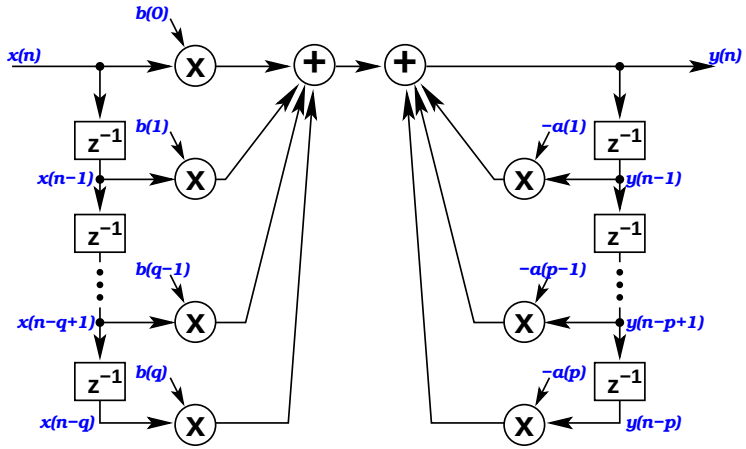
Notice that if $y(n) = x(n-1)$ then $Y(z) = z^{-1}X(z)$



So we can represent block diagrams in this fashion.

Block diagram of ARMA

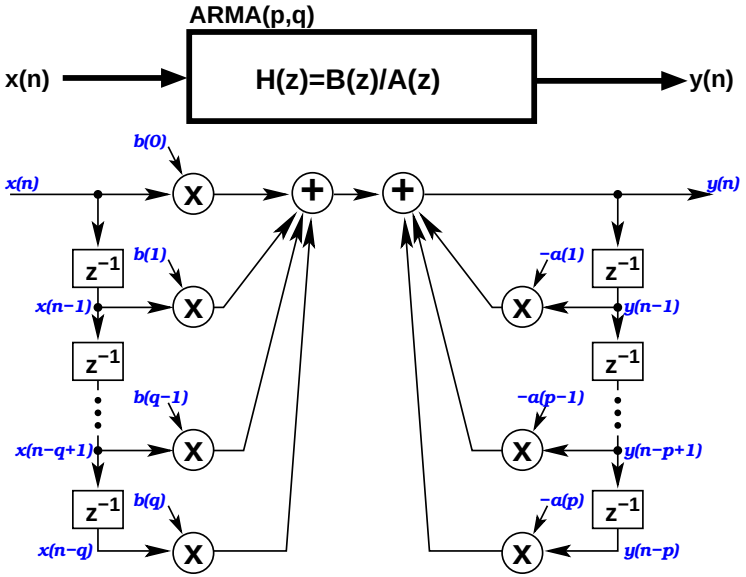
ARMA: $y(n) = -\sum_{i=1}^{p} a(i)y(n-i) + \sum_{i=0}^{q} b(i)x(n-i)$



Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.7/64

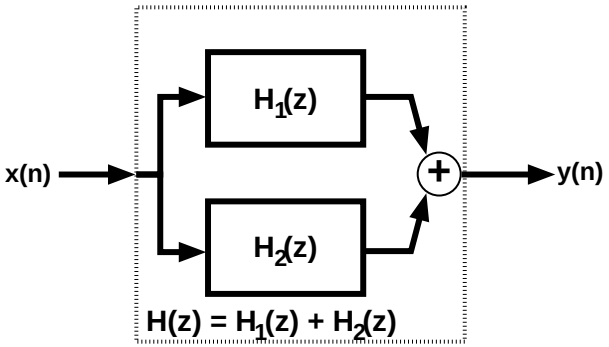
Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.8/64

Abstracting the blocks



Connecting the blocks

In parallel



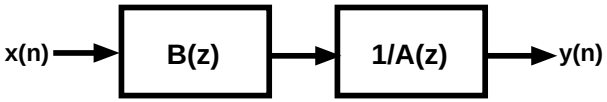
$$Y(z) = [H_1(z) + H_2(z)]X(z)$$

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.9/64

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.10/64

Connecting the blocks

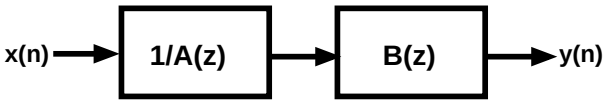
Series, or cascade



$$Y(z) = \frac{1}{A(z)} \times B(z) \times X(z)$$

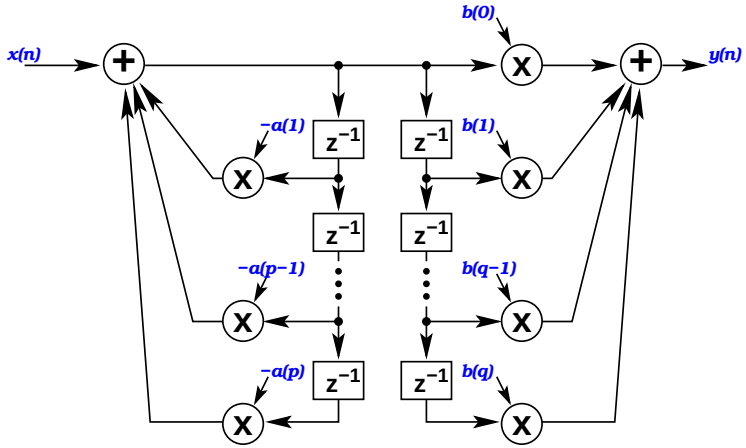
$$= \frac{B(z)}{A(z)} X(z)$$

$$Y(z) = B(z) \times \frac{1}{A(z)} \times X(z)$$



Transposed Block diagram of ARMA

ARMA: $y(n) = -\sum_{i=1}^{p} a(i)y(n-i) + \sum_{i=0}^{q} b(i)x(n-i)$



Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.11/64

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.12/64

Connecting the blocks

How do we use cascades?

- Can break a high order IIR filter into a series of cascades (of small order)
- Equivalent to factorizing the polynomial $A(z)$, e.g.

$$H(z) = B(z)\frac{1}{A(z)} = B(z)\prod_{i=1}^{n}\frac{1}{A_i(z)}$$

- can do similarly to FIR part $B(z)$
- note that degree of subparts sums to degree of combination

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.13/64

Filtering in the real world

Some things to be careful about

- **Coefficient quantization:** coefficients $a(i)$, $b(i)$ are also quantized. This shifts filter poles and zeros, possibly causing instability, or other problems.
- **Overflow:** intermediate values may overflow dynamic range, resulting in clipping, even if input and output lie within dynamic range
- use cascades to make design easier (and control of above effects)

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.14/64

Upsampling again

DFT properties: similarity
We can interleave a sequence with zeros, e.g.

$$y(n) = \begin{cases} x(n/K), & \text{if } n = 0, K, 2K, \ldots, (N-1)K \\ 0, & \text{otherwise} \end{cases}$$

The resulting DFT is

$$\mathcal{F}\{y\} = Y(k) = \begin{cases} X(k) & k = 0, \ldots, N-1 \\ X(k-N) & k = N, \ldots, 2N-1 \\ \vdots & \\ X(k-(K-1)N) & k = (K-1)N, \ldots, KN-1 \end{cases}$$

Upsampling again

Practical use: upsampling (interpolation)

We have a sequence sampled every t_s seconds, e.g. at a rate $f_s = 1/t_s$, but we need a sequence sampled at rate $K f_s$.

Approach: produce a new sequence with $K-1$ zeros interleaved between each original data point.

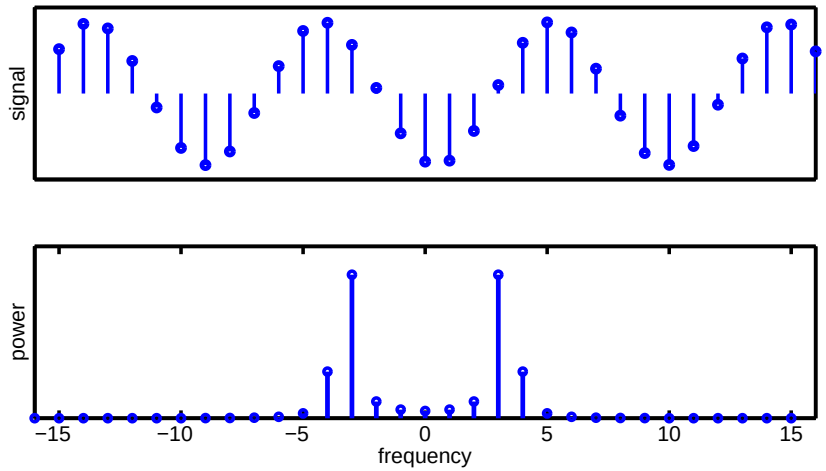
- the frequency resolution doesn't change, but now we have K repeats of the original spectrum.
- to get a signal with the same original band-limited power-spectrum, we apply a low-pass filter, smoothing the data.

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.15/64

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.16/64

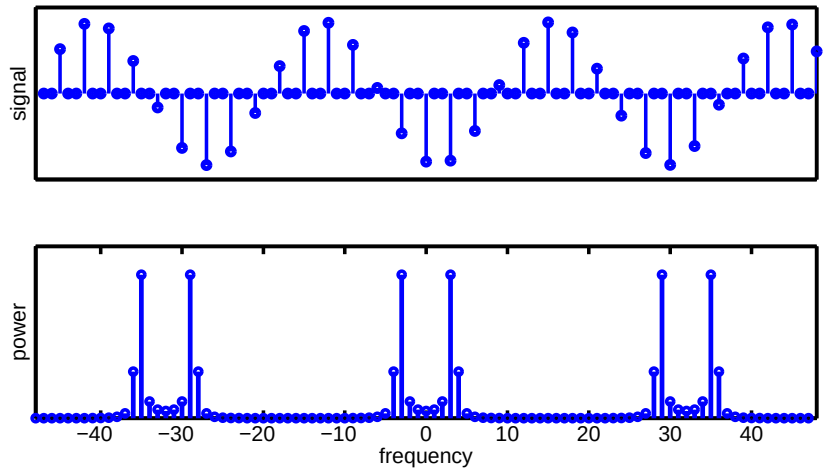
Upsampling example

32 samples (frequency 3.4 cycles)



Upsampling example

3 ×'s upsampled (96 samples)



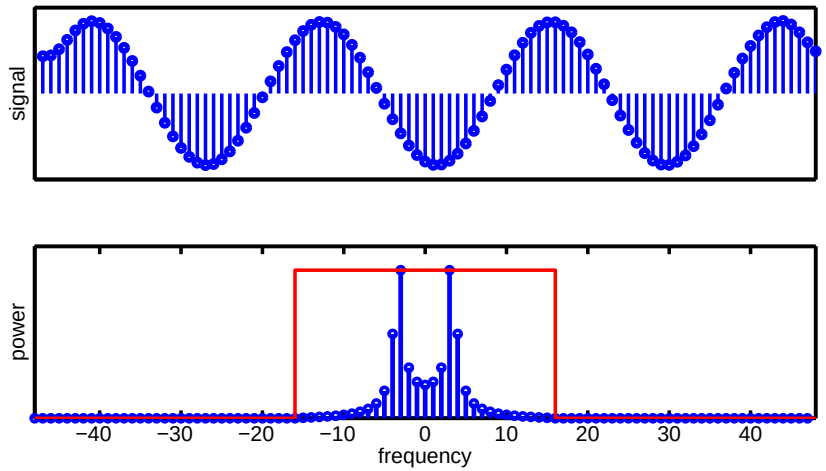

```
% file:       upsampling_1.m, (c) Matthew Roughan, Thu Aug  5 2004
%
N = 32;
x = (1:N)/N;
f1 = 3.4;
y1 = sin(2*pi*f1*x);

figure(1)
subplot(2,1,1);
hold off
plot([x; x], [zeros(size(y1)); y1],'b', 'linewidth', 3);
hold on
plot(x, y1, 'bo', 'linewidth', 4);
set(gca, 'linewidth', 3, 'fontsize', 18);
ylabel('signal');
set(gca, 'ylim', [-1.2 1.2], 'xlim', [0 max(x)], 'xtick', [], 'ytick', []);

z1 = fftshift(abs(fft(y1)).^2);
subplot(2,1,2);
hold off
plot([-N/2:N/2-1; -N/2:N/2-1], [zeros(1, N); z1], 'b', 'linewidth', 4);
hold on
plot(-N/2:N/2-1, z1, 'bo', 'linewidth', 3);
set(gca, 'linewidth', 3, 'fontsize', 18);
set(gca, 'xlim', [-N/2 N/2], 'ytick', []);
ylabel('power');
xlabel('frequency');

set(gcf, 'PaperUnits', 'centimeters', 'PaperPosition', [0 0 30 15])
print('-depsc', 'Plots/upsampling.eps');
```



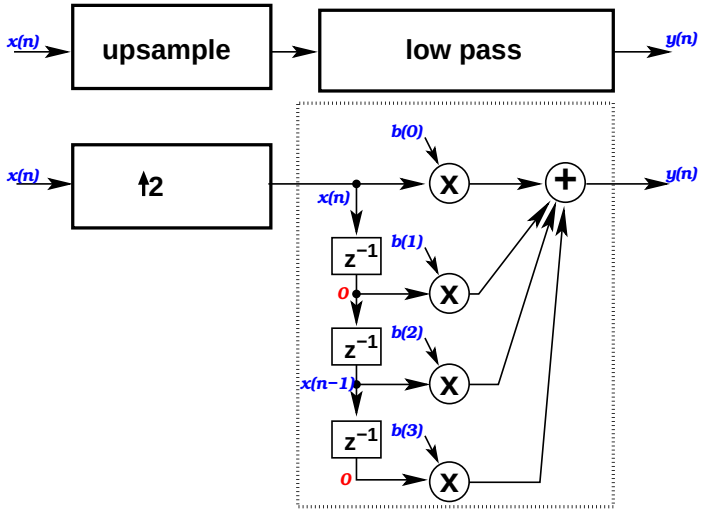
```
% file:       upsampling_2.m, (c) Matthew Roughan, Thu Aug  5 2004
%
N = 32;
x = (1:N)/N;
f1 = 3.4;
y1 = sin(2*pi*f1*x);
K = 3; % upsample by 3
upsample = zeros(1,length(y1)*K);
upsample(K*(1:length(y1))) = y1;
x_u = (1:K*N)/(K*N);

figure(2)
subplot(2,1,1);
hold off
plot([x_u; x_u], [zeros(size(upsample)); upsample],'b', 'linewidth', 3);
hold on
plot(x_u, upsample, 'bo', 'linewidth', 4);
set(gca, 'linewidth', 3, 'fontsize', 18);
ylabel('signal');
set(gca, 'ylim', [-1.2 1.2], 'xlim', [0 max(x)], 'xtick', [], 'ytick', []);

z2 = fftshift(abs(fft(upsample)).^2);
subplot(2,1,2);
hold off
plot([-K*N/2:K*N/2-1; -K*N/2:K*N/2-1], [zeros(1, K*N); z2], 'b', 'linewidth', 4);
hold on
plot(-K*N/2:K*N/2-1, z2, 'bo', 'linewidth', 3);
set(gca, 'linewidth', 3, 'fontsize', 18);
set(gca, 'xlim', [-K*N/2 K*N/2], 'ytick', []);
ylabel('power');
xlabel('frequency');

set(gcf, 'PaperUnits', 'centimeters', 'PaperPosition', [0 0 30 15])
```

Upsampling example

low pass filter, then IDFT



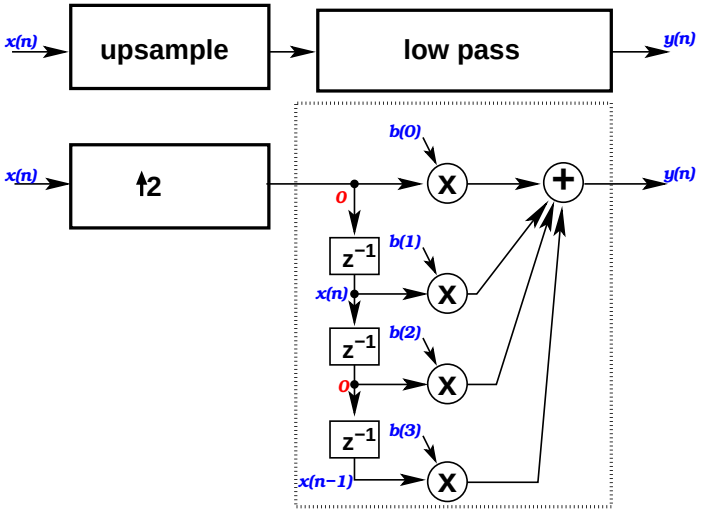
Upsampling block diagram



Notice lots of multiplies by zero


```
% file:       upsampling_3.m, (c) Matthew Roughan, Thu Aug  5 2004
%
N = 32;
x = (1:N)/N;
f1 = 3.4;
y1 = sin(2*pi*f1*x);
K = 3;
upsample = zeros(1,length(y1)*K);
upsample(K*(1:length(y1))) = y1;
x_u = (1:K*N)/(K*N);
z = fft(upsample);
z(N/2+1:end-N/2+1) = 0; %%% filter the data (using perfect low-pass in freq. domain).
y3 = real(ifft(z));

figure(3)
subplot(2,1,1);
hold off
plot([x_u; x_u], [zeros(size(y3)); y3],'b', 'linewidth', 3);
hold on
plot(x_u, y3, 'bo', 'linewidth', 4);
set(gca, 'linewidth', 3, 'fontsize', 18);
ylabel('signal');
set(gca, 'ylim', [-1.2 1.2]/K, 'xlim', [0 max(x)], 'xtick', [], 'ytick', []);

subplot(2,1,2);
hold off
plot([-K*N/2:K*N/2-1; -K*N/2:K*N/2-1], [zeros(1, K*N); abs(fftshift(z))], 'b', 'linewidth', 4);
hold on
plot(-K*N/2:K*N/2-1, abs(fftshift(z)), 'bo', 'linewidth', 3);
YY = get(gca,'ylim');
plot([-K*N/2 -N/2 -N/2 N/2 N/2 K*N/2], [0 0 max(abs(z)) max(abs(z)) 0 0], 'r', 'linewidth', 3);
set(gca, 'linewidth', 3, 'fontsize', 18);
set(gca, 'xlim', [-K*N/2 K*N/2], 'ytick', []);
```

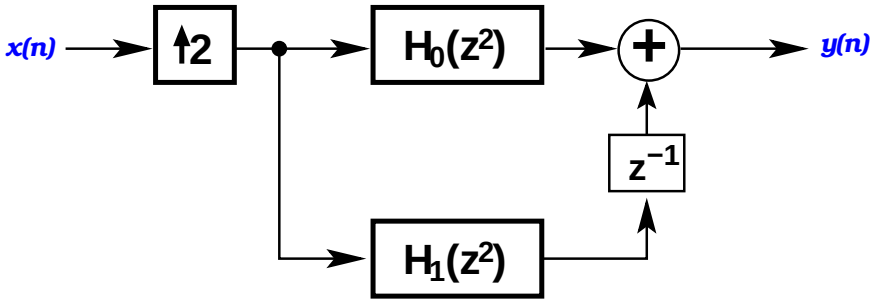
Upsampling block diagram



Notice lots of multiplies by zero

Upsampling block diagram

Trick 1: separate into two separate filters.

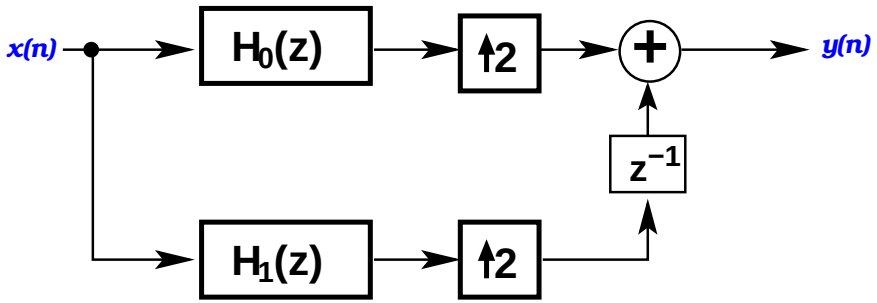


Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.21/64

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.22/64

Upsampling block diagram

Trick 2: low-pass before upsampling.



Downsampling

We can downsample by simply dropping data points, but this could cause aliasing (as the new critical frequency will depend on the downsampled rate).

As with Analogue to Digital conversion we must low-pass the signal before downsampling.

We can pull the same trick (as with upsampling) to filter at the lower rate after downsampling.

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.23/64

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.24/64

Resampling

Can do resampling by rational numbers q/p through upsampling by p, and then downsampling by q, but this is inefficient, and there are better approaches.

2D Filters

We can extend our work on filters into 2 dimensions

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.25/64

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.26/64

Convolution in 2D

Convolution generalizes to 2D, e.g., the two-dimensional convolution of continuous functions $f(x,y) = \delta(x)r(y)$ with $g(x,y) = r(x)\delta(y)$ is

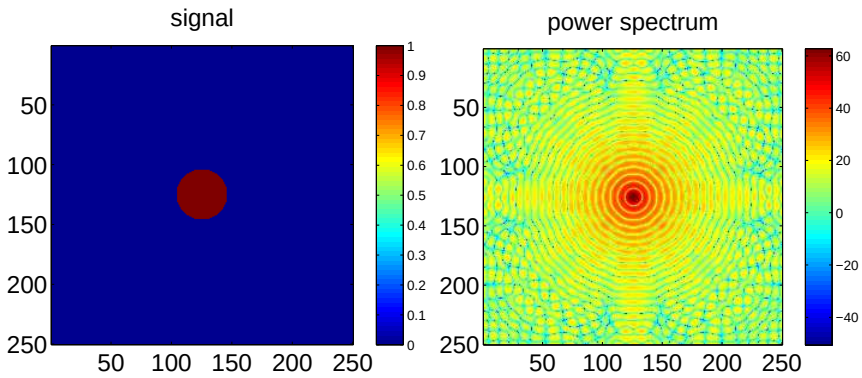
$$[f * g](x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',y')g(x-x',y-y')\,dx'\,dy'$$

Likewise, for discrete signals we can extend the idea of a LTI filter (a convolution) to 2D

$$[x * y](n,m) = \sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} x(k,l)y(n-k,m-l)$$

Filters in 2D

2D data



For finite signals we have the same issues that we have in 1D signals, of what to do at the edge of the signal (truncate, of do a circular convolution), but it is often more of an issue in a 2D signal (e.g. an image) because for an image with the same number of data points as a 1D signal, the size of the image in any one direction will be smaller. For example, if we compare a 1D signal with a 100,000 data points to an image, the image will be 100×100, so the edge effects will be more significant.

Another difference between 1D and 2D filters is that causal no longer has a clear meaning in 2D, and in fact, it is much more common to use symmetric filters in image processing (than in 1D signal processing).

However, most of the other properties of filters still apply in 2D, in particular, the convolution theorem still works.


```
% file:       two_d_filter_ex_i.m, (c) Matthew Roughan, Sat Aug 21 2004
%
N = 250;
x = (1:N)/N;
[X, Y] = meshgrid(x,x);
A = ((X-0.5).^2 + (Y-0.5).^2) < 0.007;      %  a simple image
B = 10*log10(abs(fftshift(fft2(A))).^2);    %  and its DFT

figure(1)
image(A, 'CDataMapping', 'scaled');
set(gca, 'xtick', [0:50:250], 'ytick', [0:50:250]);
set(gca, 'fontsize', 18);
title('signal');
colorbar;
set(gcf, 'PaperUnits', 'centimeters', 'PaperPosition', [0 0 12 10])
print('-depsc', sprintf('Plots/2d_filter_1a.eps', i));

figure(2)
image(B, 'CDataMapping', 'scaled');
% colormap(gray);
set(gca, 'fontsize', 18);
title('power spectrum');
colorbar;
set(gcf, 'PaperUnits', 'centimeters', 'PaperPosition', [0 0 12 10])
print('-depsc', sprintf('Plots/2d_filter_1b.eps', i));
```

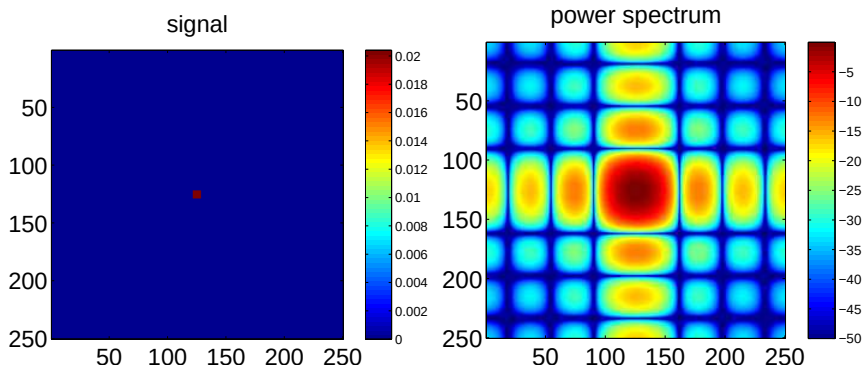
Filters in 2D

Spatial rectangular low-pass

$$\frac{1}{49}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Filters in 2D

Spatial rectangular low pass, frequency response



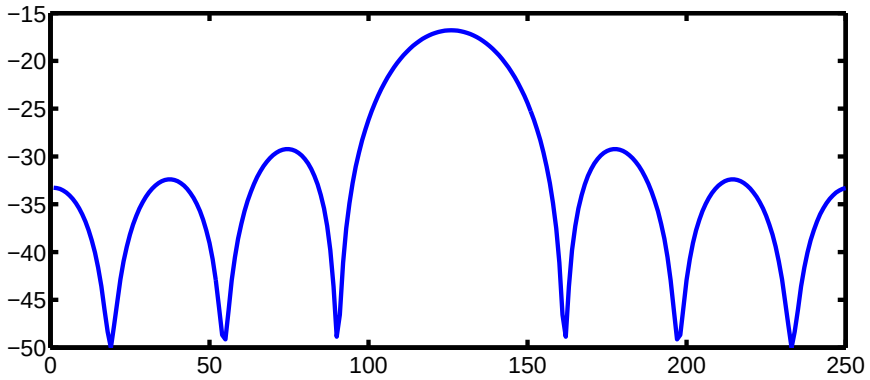
Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.29/64

An image showing the filter (from Slide 4) and its FT.

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.30/64

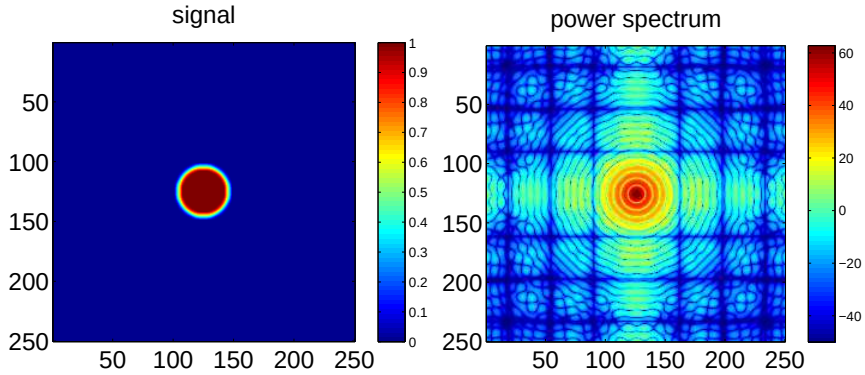
Filters in 2D

Spatial rectangular low pass, frequency response 1D



Filters in 2D

Spatial rectangular low pass




```
% file:       two_d_filter_ex_iv.m, (c) Matthew Roughan, Sat Aug 21 2004
%
N = 250; X = zeros(N,N);
X(125,125) = 1;
M = 7;
D = filter2(ones(M,M)/M^2, X, 'same');
C = 10*log10(abs(fftshift(fft2(D))).^2);
C(find(C < -50)) = -50;

figure(4)
image(D, 'CDataMapping', 'scaled');
set(gca, 'xtick', [0:50:250], 'ytick', [0:50:250]);
set(gca, 'fontsize', 18);
title('signal');
colorbar;
set(gcf, 'PaperUnits', 'centimeters', 'PaperPosition', [0 0 12 10])
print('-depsc', sprintf('Plots/2d_filter_4a.eps', i));

figure(4)
image(C, 'CDataMapping', 'scaled');
set(gca, 'fontsize', 18);
title('power spectrum');
colorbar;
set(gcf, 'PaperUnits', 'centimeters', 'PaperPosition', [0 0 12 10])
print('-depsc', sprintf('Plots/2d_filter_4b.eps', i));

figure(5)
plot(mean(C), 'linewidth', 3);
set(gca, 'fontsize', 16, 'linewidth', 3);
set(gcf, 'PaperUnits', 'centimeters', 'PaperPosition', [0 0 25 10])
print('-depsc', sprintf('Plots/2d_filter_5.eps', i));
```


The image is the convolution of the image on page 3, and the filter shown on Slides 4-5.

As expected the FT is the product of the spectrums of the image on slide 3, and the filter whose FT is shown on Slide 5.

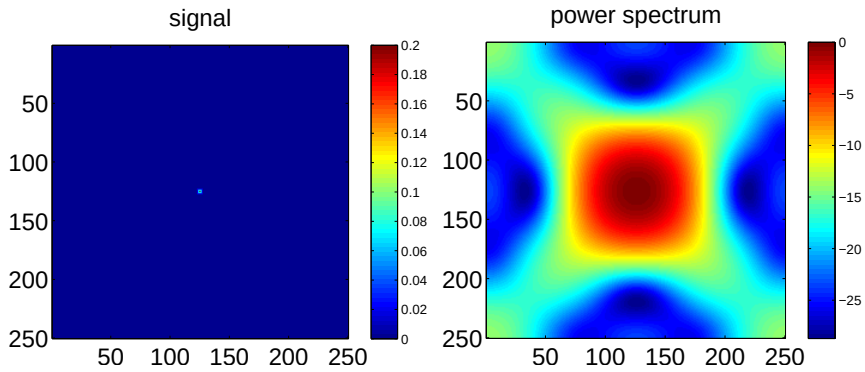
Filters in 2D

Approximately Gaussian low-pass

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 4 & 4 & 4 & 1 \\ 1 & 4 & 12 & 4 & 1 \\ 1 & 4 & 4 & 4 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Filters in 2D

Approximately Gaussian low-pass, frequency response



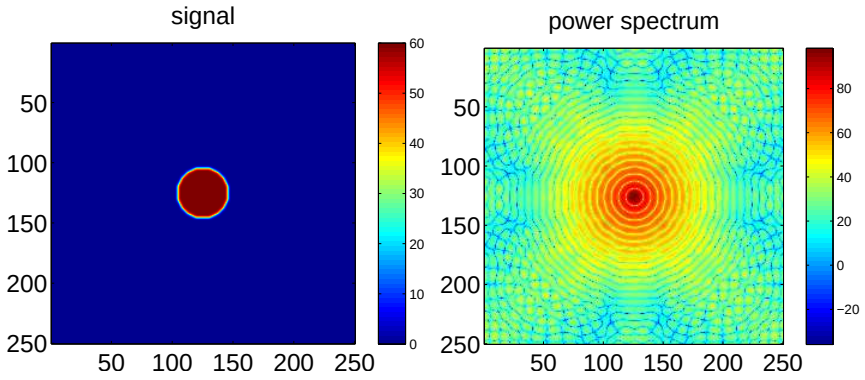
Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.33/64

Image shows the approximately Gaussian low-pass from Slide 8, and its FT.

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.34/64

Filters in 2D

Approximately Gaussian low-pass



Edge detection

Edge detection is a high pass operation, but there are a number of ways to do this in 2D.

- gradient: look for max and min in gradients in image
  - Sobel
  - Prewitt
  - Roberts
- Laplacian: look for zeros of second derivative
  - Marrs-Hildreth

Convolution of the image (Slide 3) and the Gaussian filter from Slides 8-9.

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.35/64

See also:
`http://www.owlnet.rice.edu/~elec539/Projects97/morphjrks/moredge.html`

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.36/64

Sobel Edge detection

Look for vertical and horizontal edge separately, using filters

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

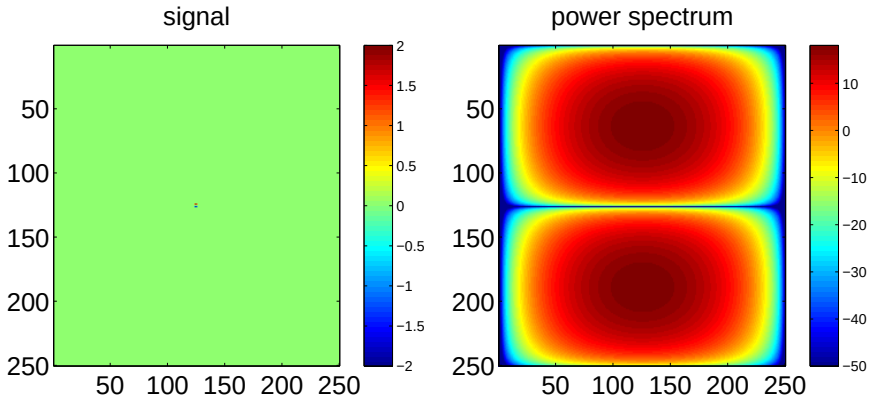
The threshold for values above or below T.

Alternatively, can combine to get edge magnitude and direction by

$$\begin{aligned} M_{\text{sobel}} &= \sqrt{M^2_{\text{vertical}} + M^2_{\text{horizontal}}} \\ \phi_{\text{sobel}} &= \tan^{-1}\left(M_{\text{vertical}}/M_{\text{horizontal}}\right) \end{aligned}$$

Filters in 2D

Sobel edge detection, freq. response



Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.37/64

The filter being use

$$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

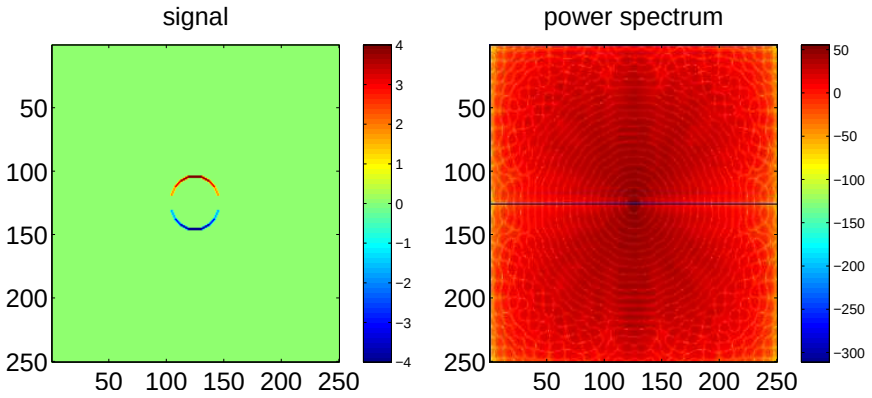
We can't see the filter co-efficients in the left image (because of the resolution), but we can see the frequency response.

We can see that the filter is a low-pass in the horizontal direction, but it acts more like a high-pass (actually a band-pass) in the vertical direction.

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.38/64

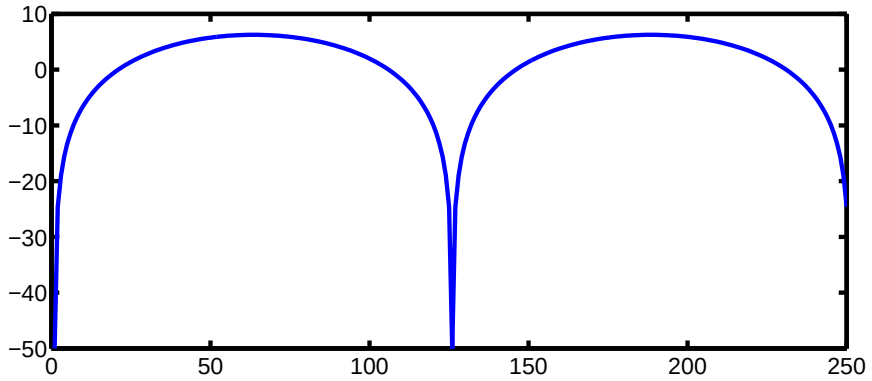
Filters in 2D

Sobel edge detection applied to image.



Filters in 2D

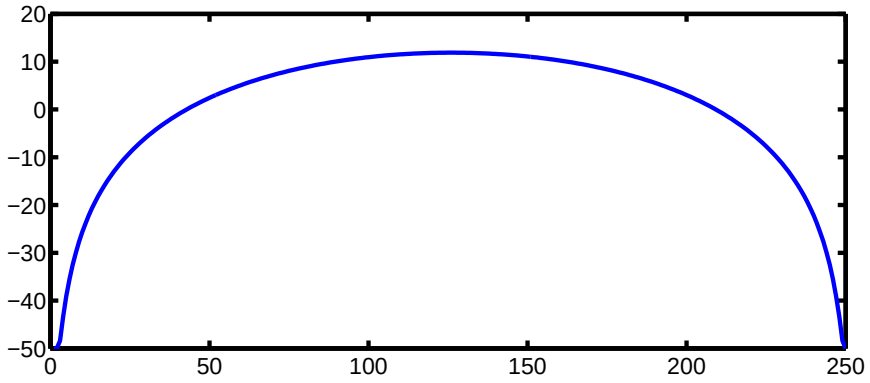
Vertical high pass (Sobel edge detection), freq. response, horizontal




```
% file:       two_d_filter_ex_vi.m, (c) Matthew Roughan, Sat Aug 21 2004
%                 Sobel filter
N = 250;
x = (1:N)/N;
[X, Y] = meshgrid(x,x);
A = ((X-0.5).^2 + (Y-0.5).^2) < 0.007;
V = [[-1 -2 -1];[0 0 0];[1 2 1]];
H = [[-1 0 1];[-2 0 2];[-1 0 1]];
Dv = filter2(V, A, 'same');
Cv = 10*log10(abs(fftshift(fft2(Dv))).^2);
Cv(find(C < -50)) = -50;

figure(6)
subplot(1,2,1)
image(Dv, 'CDataMapping', 'scaled');
set(gca, 'xtick', [0:50:250], 'ytick', [0:50:250]);
set(gca, 'fontsize', 18);
title('signal');
colorbar;

figure(6)
subplot(1,2,2)
image(Cv, 'CDataMapping', 'scaled');
set(gca, 'fontsize', 18);
title('power spectrum');
colorbar;

set(gcf, 'PaperUnits', 'centimeters', 'PaperPosition', [0 0 25 10])
print('-depsc', sprintf('Plots/2d_filter_6.eps', i));
```

Filters in 2D

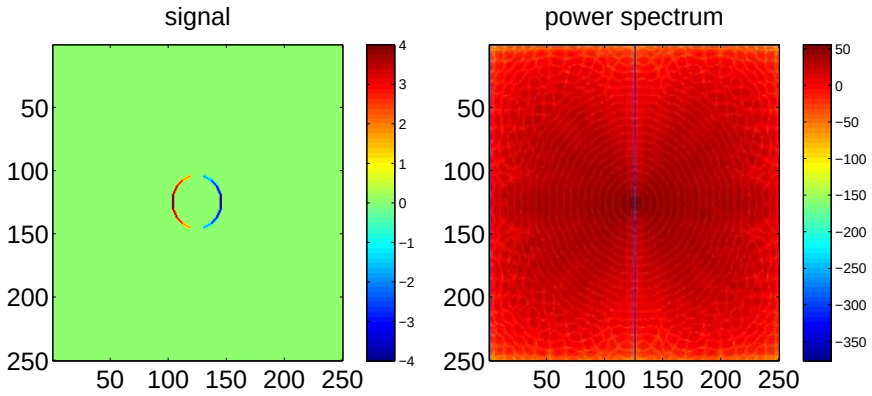
Vertical high pass (Sobel edge detection), freq. response, vertical



Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.41/64

Filters in 2D

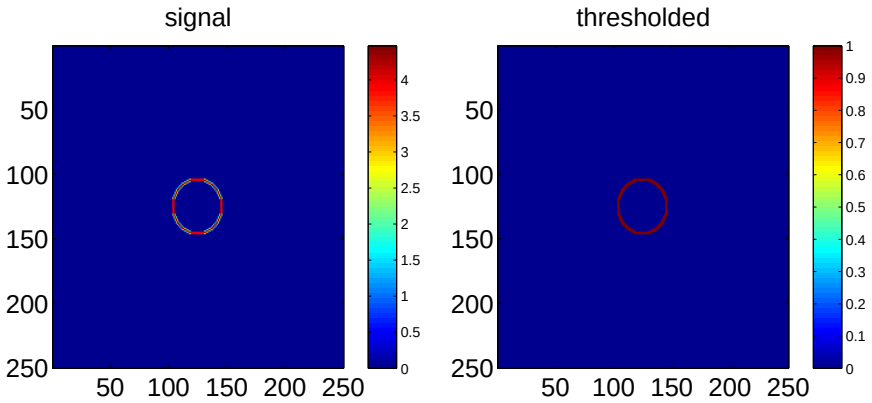
Horizontal high pass (Sobel edge detection)



Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.42/64

Filters in 2D

Combined Sobel filters, and thresholded version



Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.43/64

Prewitt filters

Look for vertical and horizontal edge separately, using filters

$$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.44/64

Roberts filters

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.45/64

Laplacian

Approximation of $\nabla^2 = \dfrac{\partial^2}{\partial x^2} + \dfrac{\partial^2}{\partial y^2}$

3×3

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

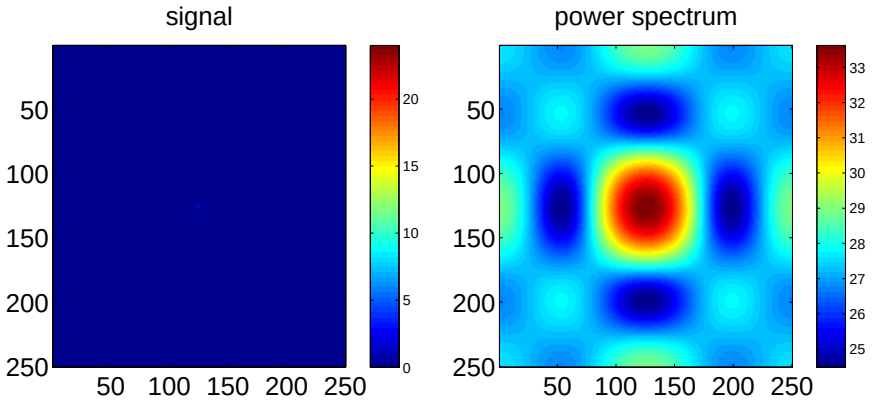
5×5

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 24 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.46/64

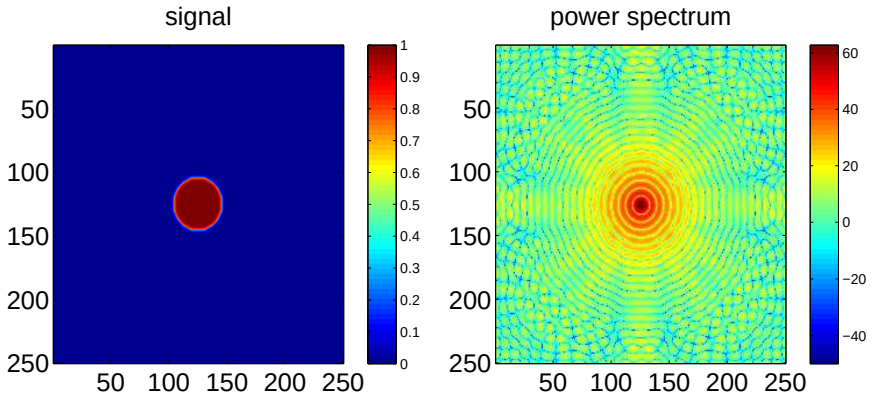
Filters in 2D

5×5 Laplacian, and its frequency response



Filters in 2D

5×5 Laplacian applied to image




```
% file:       two_d_filter_ex_51.m, (c) Matthew Roughan, Mon Jul 24 2006
%                   Laplacian edge detection
N = 250; X = zeros(N,N);
X(125,125) = 1;
A = ((X-0.5).^2 + (Y-0.5).^2) < 0.007;
K = [[1 1 1 1 1]
     [1 1 1 1 1]
     [1 1 24 1 1]
     [1 1 1 1 1]
     [1 1 1 1 1]];
D = filter2(K, X, 'same');
C = 10*log10(abs(fftshift(fft2(D))).^2);
C(find(C < -50)) = -50;

figure(50)
subplot(1,2,1)
image(D, 'CDataMapping', 'scaled');
set(gca, 'xtick', [0:50:250], 'ytick', [0:50:250]);
set(gca, 'fontsize', 18);
title('signal');
colorbar;

figure(50)
subplot(1,2,2)
image(C, 'CDataMapping', 'scaled');
set(gca, 'fontsize', 18);
title('power spectrum');
colorbar;

set(gcf, 'PaperUnits', 'centimeters', 'PaperPosition', [0 0 25 10])
print('-depsc', sprintf('Plots/2d_filter_50.eps', i));
```

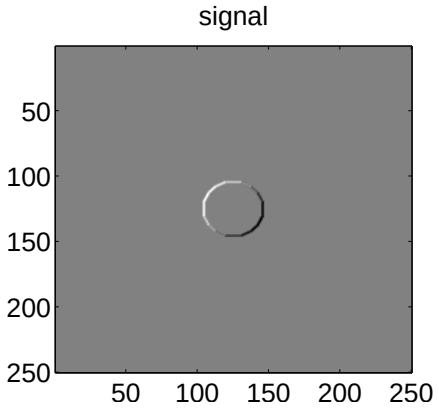


```
% file:       two_d_filter_ex_51.m, (c) Matthew Roughan, Mon Jul 24 2006
%                   Laplacian edge detection
N = 250;  x = (1:N)/N;
[X, Y] = meshgrid(x,x);
A = ((X-0.5).^2 + (Y-0.5).^2) < 0.007;
K = [[1 1 1  1 1]
     [1 1 1  1 1]
     [1 1 24 1 1]
     [1 1 1  1 1]
     [1 1 1  1 1]];
D = filter2(K/sum(sum(K)), A, 'same');
C = 10*log10(abs(fftshift(fft2(D))).^2);
C(find(C < -50)) = -50;

figure(51)
subplot(1,2,1)
image(D, 'CDataMapping', 'scaled');
set(gca, 'xtick', [0:50:250], 'ytick', [0:50:250]);
set(gca, 'fontsize', 18);
title('signal');
colorbar;

figure(51)
subplot(1,2,2)
image(C, 'CDataMapping', 'scaled');
set(gca, 'fontsize', 18);
title('power spectrum');
colorbar;

set(gcf, 'PaperUnits', 'centimeters', 'PaperPosition', [0 0 25 10])
print('-depsc', sprintf('Plots/2d_filter_51.eps', i));
```

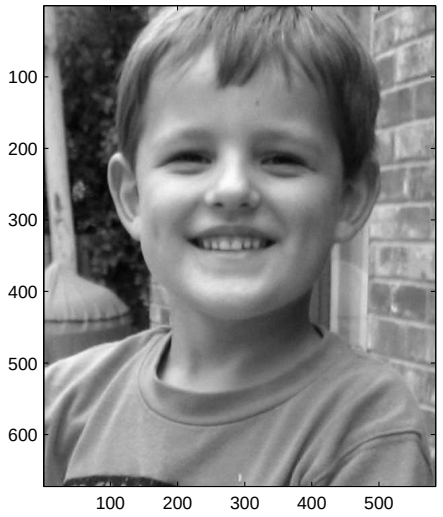
Embossing an image

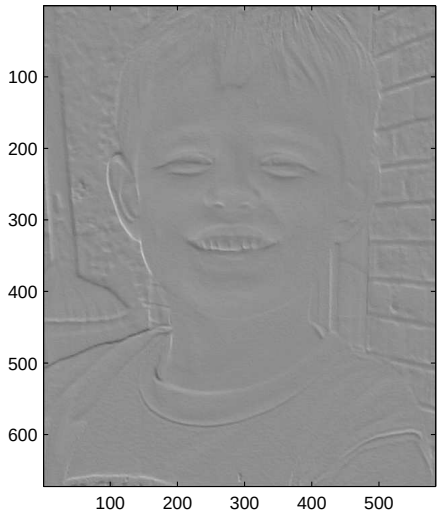
Fancy effects from simple filters, e.g. take $\theta = \pi/6$ and

$$\text{image} = M_{\text{sobel}}^{\text{horizontal}} \cos(\theta) + M_{\text{sobel}}^{\text{vertical}} \sin(\theta)$$



Embossing an image





A surprising number of image effects (e.g. in Photoshop or the Gimp) are simply combinations of filters.


```
% file:       two_d_filter_ex.m, (c) Matthew Roughan, Sat Aug 21 2004
%
image1 = 'james.png';
[i1,map1] = imread(image1, 'png');
V = [[-1 -2 -1];[0 0 0];[1 2 1]];
H = [[-1 0 1];[-2 0 2];[-1 0 1]];
Dh = filter2(H, i1, 'same');
Dv = filter2(V, i1, 'same');

figure(100)
subplot(1,2,1)
image(i1, 'CDataMapping', 'scaled');

subplot(1,2,2)
theta = -pi/3;
emboss_image = (Dh*cos(theta) + Dv*sin(theta));
image(emboss_image, 'CDataMapping', 'scaled');
opts = struct('height',10, 'Color', 'bw');
exportfig(gcf,'Plots/2d_filter_emboss.eps', opts, 'color', 'gray');
```

Tomography

One of the more fantastic achievements of signal processing is in the field of medical imaging where tomography is used to see inside a living body. The underlying techniques rely on transforms, and have wide applications in many other areas.

Applications: tomography

- in many problems we **can't** directly observe object
- we observe indirect measurements
  - seismology (determining size/shape of ore body)
  - oceanography (acoustic observations to get water temp)
  - archaeology (e.g. finding remains with GPR)
  - medical imaging (CT, MRI, PET scans)
  - manufacturing (process monitoring)
  - networks (traffic matrix estimation)
- inverse problem to take indirect measurements and infer true state from these
- tomography (from 'cuts' = tomo-)

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.51/64

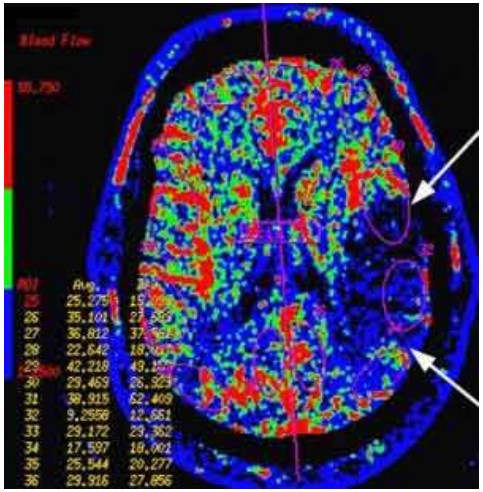
For notes on tomography using FT (and Radon transforms) see Bracewell, p.356.
Other references:
http://www.owlnet.rice.edu/~elec301/Projects00/tomography/R_intro.htm

In particular areas:
Ocean Acoustic Tomography http://www.oal.whoi.edu/tomo2.html
Archaeology http://archaeology.huji.ac.il/ct/
Medical Imaging http://www.triumf.ca/welcome/petscan.html Manufacturing http://www.tomography.umist.ac.uk/intro.shtml
Seismology http://www.itso.ru/GEOTOMO/paper_moscow2003/index.html
Traffic Matrices http://internal.maths.adelaide.edu.au/people/mroughan/traffic_matrices.html

Matlab code:
http://www.owlnet.rice.edu/~elec301/Projects00/tomography/code.htm

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.52/64

Computed Aided Tomography

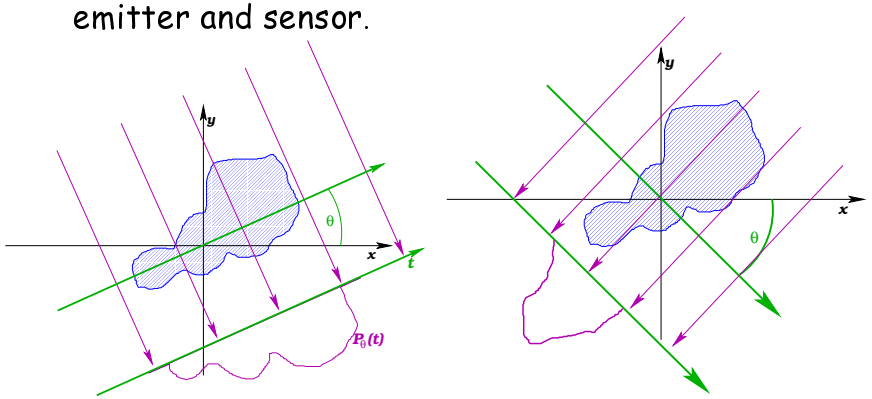


http://www.radiologyinfo.org/photocat/photos.cfm?image=hd-ct-perfusion.jpg
&&subcategory=Head&&stop=9

Computed Aided Tomography

Simple view

- take a series of observations from different angles
- at each angle, measure density of material between emitter and sensor.

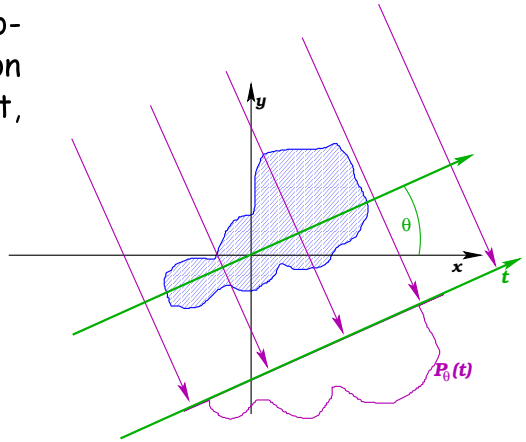


Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.53/64

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.54/64

Computed Aided Tomography

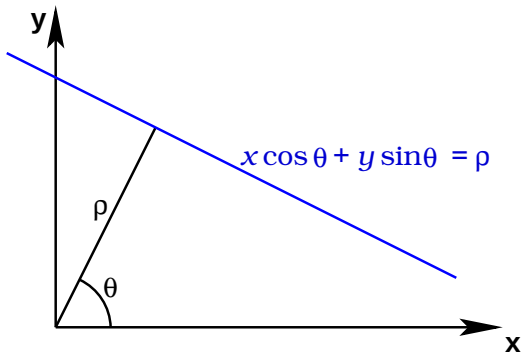
We can compute the projection $P_\theta(t)$ using the Radon transform of the object, e.g.



$$P_\theta(t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,\delta(t - x\cos\theta - y\sin\theta)\,dx\,dy$$

An example: Radon transform

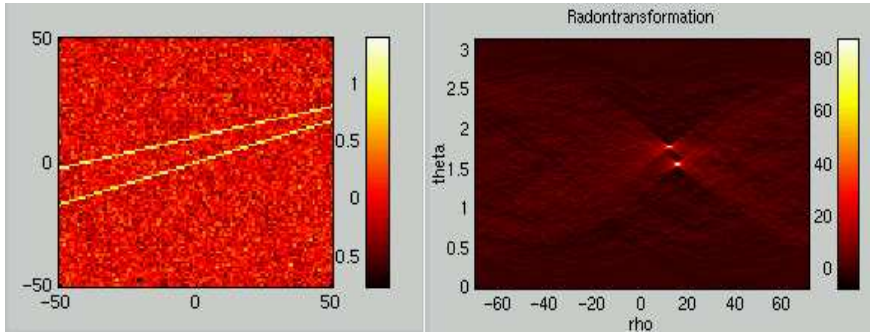
$$F(\rho,\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,\delta(\rho - x\cos\theta - y\sin\theta)\,dx\,dy$$



Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.55/64

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.56/64

An example: Radon transform



http://eivind.imm.dtu.dk/staff/ptoft/Radon/Radon.html

Fourier transform of the cut

We can compute the FT of a slice

$$
\begin{aligned}
F_\theta(s) &= \int_{-\infty}^{\infty} P_\theta(t) e^{-i2\pi st}\, dt \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,\delta(t - x\cos\theta - y\sin\theta)\, dx\, dy\, e^{-i2\pi st}\, dt \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y) \int_{-\infty}^{\infty} \delta(t - x\cos\theta - y\sin\theta) e^{-i2\pi st}\, dt\, dx\, dy \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y) e^{-i2\pi s(x\cos\theta + y\sin\theta)}\, dx\, dy \\
&= F(s\cos\theta, s\sin\theta)
\end{aligned}
$$

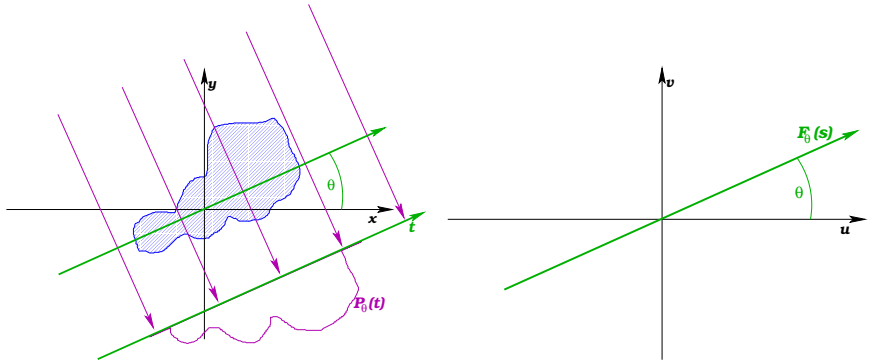
Links to Radon transform:
http://mathworld.wolfram.com/RadonTransform.html
http://eivind.imm.dtu.dk/staff/ptoft/Radon/Radon.html
http://www.anc.ed.ac.uk/~amos/hough.html

Fourier slice theorem

The Fourier transform of a projection (slice) through the object $F_\theta(s)$

$$F_\theta(s) = F(s\cos\theta, s\sin\theta)$$

where $F(u,v)$ is the 2D Fourier transform of the object.



Notional algorithm

We can now draw a notional algorithm

- Radon transform (via slices)
- Fourier transform to get slices of FT
- inverse Fourier transform to get back to object

But this fails to take into account

- discrete measurements
- finite number of projections

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.59/64

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.60/64

Back-projection filtering algorithm

Write inverse FT polar coordinates:

$$\begin{aligned} u &= \omega\cos\theta \\ v &= \omega\sin\theta \end{aligned}$$

So $du\,dv = \omega\,d\omega\,d\theta$, and

$$\begin{aligned} f(x,y) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{i2\pi(ux+vy)}\,du\,dv \\ &= \int_{0}^{2\pi}\int_{0}^{\infty} F(\omega,\theta)e^{i2\pi\omega(x\cos\theta+y\sin\theta)}\,\omega\,d\omega\,d\theta \\ &= \int_{0}^{\pi}\int_{0}^{\infty} F(\omega,\theta)e^{i2\pi\omega(x\cos\theta+y\sin\theta)}\,\omega\,d\omega\,d\theta \\ &\quad + \int_{0}^{\pi}\int_{0}^{\infty} F(\omega,\theta+\pi)e^{i2\pi\omega(x\cos(\theta+\pi)+y\sin(\theta+\pi))}\,\omega\,d\omega\,d\theta \end{aligned}$$

Back-projection filtering algorithm

From symmetry $F(\omega,\theta+\pi) = F(-\omega,\theta)$, so

$$\begin{aligned} f(x,y) &= \int_{0}^{\pi}\int_{-\infty}^{\infty} F(\omega,\theta)e^{i2\pi\omega(x\cos\theta+y\sin\theta)}\,|\omega|\,d\omega\,d\theta \\ &= \int_{0}^{\pi}\int_{-\infty}^{\infty} F_\theta(\omega)e^{i2\pi\omega(x\cos\theta+y\sin\theta)}\,|\omega|\,d\omega\,d\theta \\ &= \int_{0}^{\pi} Q_\theta(x\cos\theta+y\sin\theta)\,d\theta \end{aligned}$$

where

$$Q_\theta(t) = \int_{-\infty}^{\infty} F_\theta(\omega)|\omega|e^{i2\pi\omega t}\,d\omega$$

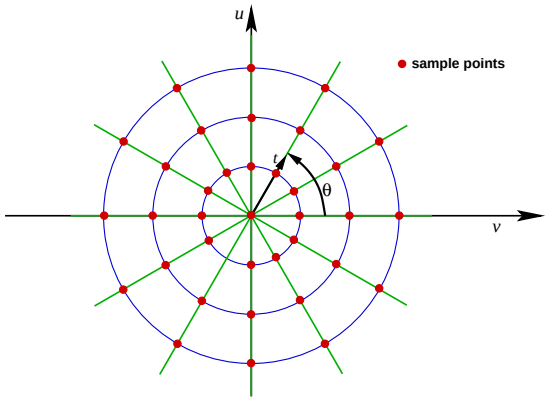
$Q_\theta(t)$ represents a filtered version of $P_\theta(t)$, with frequency response $|\omega|$.

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.61/64

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.62/64

Sampling of Fourier Domain

Assume that t and θ are sampled uniformly, the sampling of the Fourier domain looks like.

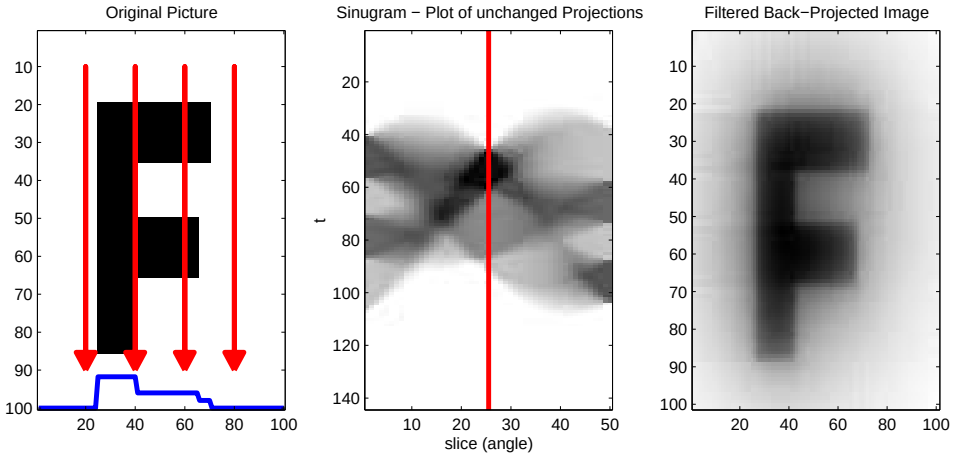


Samples are more dense around the center!

- filtering tries to counteract this

Examples

50 projections



http://www.owlnet.rice.edu/~elec301/Projects00/tomography/code.htm

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.63/64

Transform Methods & Signal Processing (APP MTH 4043): lecture 06 – p.64/64