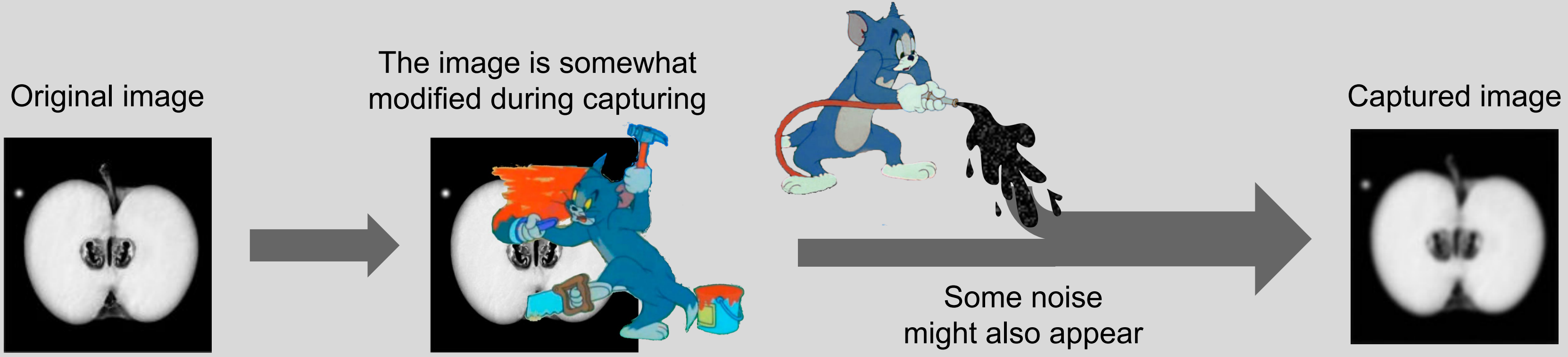
Lab 07

Basic Image Processing Algorithms
Fall 2022

Image degradation and restoration

A typical imaging system



DEGRADATION block diagram

Image degradation and restoration

DEGRADATION
block diagram

"Degradation happens"

We don't want a reduced quality image but that's the best the system can capture.

Quality loss happens due to motion, out-of-focus lenses, transmission and quantization errors etc.

Usually it is not possible (or affordable) to eliminate the causes of the degradation. Instead, we use restoration approaches to get back the original image from the low quality one.

Let's take it to the next level!

DEGRADATION block diagram

Spatial domain

$$x(n_1, n_2) \otimes h(n_1, n_2) + n(n_1, n_2) = y(n_1, n_2)$$

The system is LSI (*linear space-invariant*) so we can Fourier transform it to the frequency domain:

Frequency domain

$$X(\omega_1, \omega_2) \cdot H(\omega_1, \omega_2) + N(\omega_1, \omega_2) = Y(\omega_1, \omega_2)$$

Point spread function & Optical transfer function

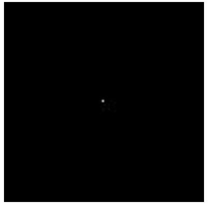
We can describe H with an image-pair. Let’s create a well-known image and send it through the imaging system. The response will be a modified image.
If we assume no noise then the input-output pair perfectly describes the H.

an arbitrary input

the response of the system

If the input image contains only one point (a pixel) then we call the response the *point spread function* and we can use this single image to describe the system.

a “point”



the response is the
point spread function

Point spread function & Optical transfer function

The point spread function can be used to realize the system. An image convolved with the point spread function is the response of the *H* block.

Spatial domain

an arbitrary input | convolved with | the PSF | is | the response of the system

$\otimes$

$=$

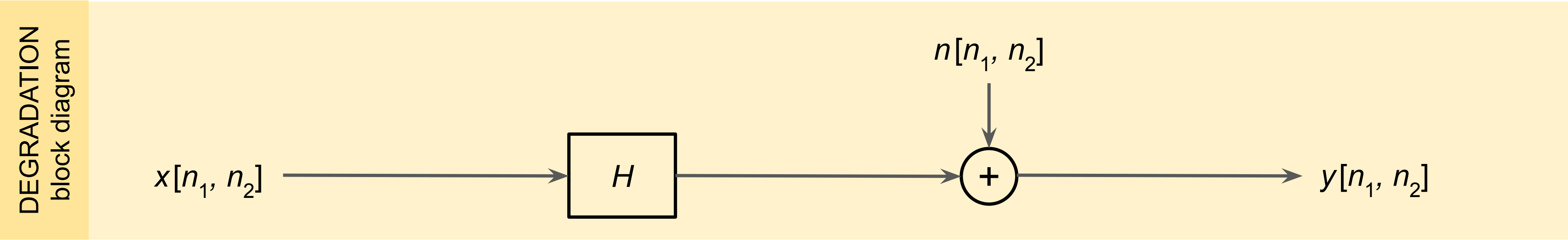
$$x(n_1, n_2) \otimes h(n_1, n_2) = y(n_1, n_2)$$

Frequency domain

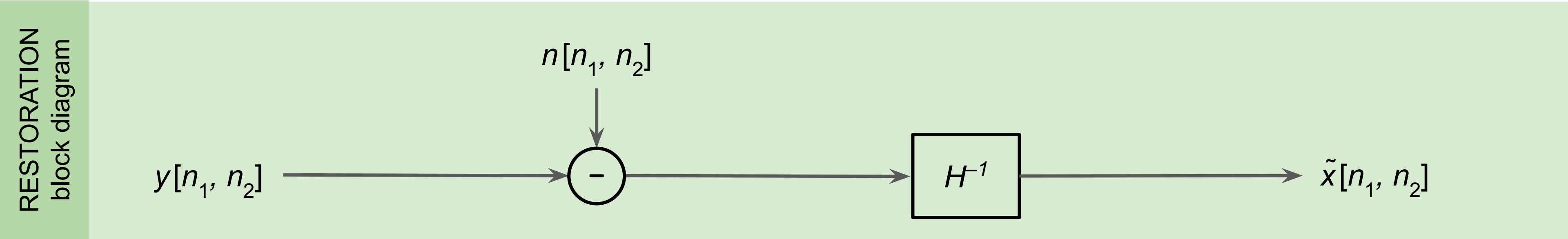
$$X(\omega_1, \omega_2) \bullet H(\omega_1, \omega_2) = Y(\omega_1, \omega_2)$$

We call the Fourier transform of the PSF "**optical transfer function**" (OTF)

Image degradation and restoration



Since we have the model of the degradation system, we can build its inverse.



Now we just have to find n and H^{-1} and we are ready :)

Image degradation and restoration

RESTORATION block diagram

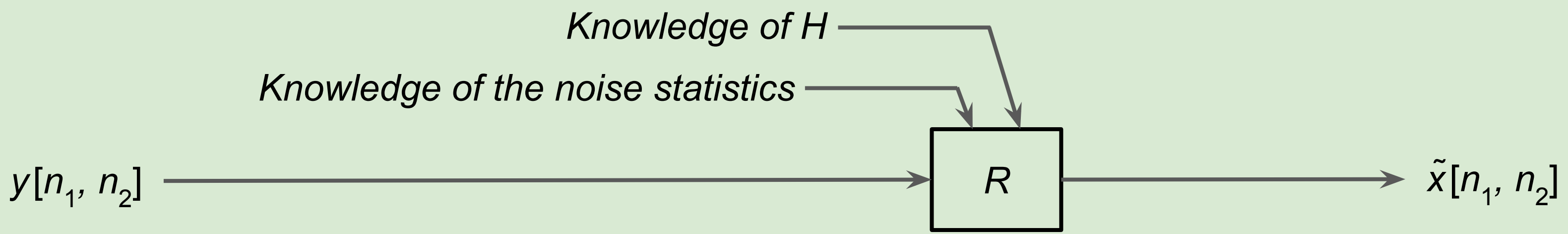
In practice it is impossible to find the exact n $\Rightarrow$ subtraction cannot be done.
However, we can do some ***noise statistics*** or ***assume*** the type of noise.

In practice we don’t know the exact H either.
However, we can do ***measurements*** or we can ***estimate*** the system. Also, we have to ***deal with the noise*** here.

Therefore, instead of using H^{-1} it is very common to create an R ‘recovery’ system.

Image degradation and restoration

RESTORATION block diagram



Depending on the restoration approach the *R* system can be…

Inverse Filter	$\underset{x}{\operatorname{argmin}}\left(\lVert y - Hx \rVert^2\right)$	$R(\omega_1, \omega_2) = \dfrac{1}{H(\omega_1, \omega_2)}$
Constrained Least Square	$\underset{x}{\operatorname{argmin}}\left(\lVert y - Hx \rVert^2 + \alpha \lVert Cx \rVert^2\right)$	$R(\omega_1, \omega_2) = \dfrac{H^*(\omega_1, \omega_2)}{\lvert H(\omega_1, \omega_2)\rvert^2 + \alpha \lvert C(\omega_1, \omega_2)\rvert^2}$
Wiener Filter	$\underset{x}{\operatorname{argmin}}\left(\mathbb{E}\left[\lVert x - \tilde{x} \rVert\right]\right)$	$R(\omega_1, \omega_2) = \dfrac{H^*(\omega_1, \omega_2)}{\lvert H(\omega_1, \omega_2)\rvert^2 + \dfrac{P_{NN}(\omega_1, \omega_2)}{P_{XX}(\omega_1, \omega_2)}}$

Now please

download the ‘Lab 07’ code package

from the

moodle system

Exercise 1

Implement the function `degradation` in which:

- The function has one input and three output arguments.
- Inputs:
  - `x` the input image (grayscale, double, within range [0,1])
- Outputs:
  - `y` the degraded output image
  - `h` the kernel of the imaging system
  - `n` the values of the additive noise layer

Exercise description continues on the next slide...

Exercise 1 – continued

The degradation kernel is a `'motion'` type kernel with parameters `len=21`, and `theta=11`. Use `fspecial` to produce the kernel matrix.

The additive noise is a white noise; the `n` should be filled with normally distributed random numbers (use `randn`). The variance should be $\sigma^2 = 0.001$ (Multiply the result of `randn` by `sqrt(0.001)` to get the desired variance.)

After creating the appropriate `h` and `n` matrices, apply the kernel to the input using `imfilter` with the options `'replicate'`, `'same'`, `'conv'`, and after this, add the noise layer to the filtered image.

$$y = x \otimes h + n$$

Execute `script1.m` and check your results!

Original input image

Degraded image (filter + noise)

Exercise 2

Implement the function `restoration_CLS` in which:

- The function has four inputs:
  - `y` is the degraded image,
  - `h` is the point spread function of the system,
  - `alpha` is a regularization parameter, and
  - `c` is the point spread function of a high pass filter
- The function has to return the restored image (`x_tilde`).

Exercise description continues on the next slide…

Exercise 2 – continued

First, move everything to the frequency domain. You have to compute the `Y` matrix (the degraded image in the frequency domain) using `fft2`.

After this, compute the optical transfer functions from the given point spread functions. For this use `psf2otf()`, where the first argument is the PSF to be converted, and the second argument is the desired size of the OTF.

In our case, we want the OTFs to be the same size as `Y`!

Exercise description continues on the next slide...

Exercise 2 – continued

After this, implement the formula for the `R` restoration matrix:

$$R(\omega_1, \omega_2) = \frac{H^*(\omega_1, \omega_2)}{|H(\omega_1, \omega_2)|^2 + \alpha|C(\omega_1, \omega_2)|^2}$$

in which H^* means complex conjugate (use the `conj()` function in MATLAB).

Finally, apply the restoration system to the input in the frequency domain. Multiply `Y` with `R` elementwise and transform back the result to the spatial domain.

$$\tilde{x} = \left|\mathrm{IDFT}\{R \cdot \mathrm{DFT}\{y\}\}\right|$$

Execute `script2.m` and check your results!

Original input image

Degraded image (filter + noise)

Restored image (CLS)

Exercise 3

Implement the function `restoration_wiener` in which:

- The function has three inputs:
  - `y` is the degraded image,
  - `h` is the point spread function of the system,
  - `n` is a noise layer (an image containing noise only)
- The function have to return the restored image (`x_tilde`).

Exercise description continues on the next slide…

Exercise 3 – continued

First, move everything to the frequency domain (just as in Exercise 2).

Moreover, you have to compute two additional values, the power spectra. According to the formula (next slide) we need the `P_XX` and `P_NN` values. Use the expression

$$P_{XX}(\omega_1, \omega_2) = X(\omega_1, \omega_2) \cdot X^*(\omega_1, \omega_2)$$

in which X* means complex conjugate (use the `conj()` function in MATLAB).

Ooops! We don't know the matrix X. (Well, it is impossible to know it, that's the point of the degradation.) Nevermind, let's approximate `P_XX` with `P_YY`.

Exercise description continues on the next slide...

Exercise 3 – continued

After this, implement the formula for the `R` restoration matrix:

$$R(\omega_1, \omega_2) = \frac{H^*(\omega_1, \omega_2)}{|H(\omega_1, \omega_2)|^2 + \frac{P_{NN}(\omega_1, \omega_2)}{P_{XX}(\omega_1, \omega_2)}}$$

in which H^* means complex conjugate (use the `conj()` function in MATLAB).

Finally, apply the restoration system to the input in the frequency domain. Multiply `Y` with `R` elementwise and transform back the result to the spatial domain.

$$\tilde{x} = \left|\mathrm{IDFT}\left\{R \cdot \mathrm{DFT}\{y\}\right\}\right|$$

Execute `script3.m` and check your results!

Original input image

Degraded image (filter + noise)

Restored image (Wiener)

Exercise 4

Implement the function `restoration_wiener_white` in which:

- The function has three inputs:
  - `y` is the degraded image,
  - `h` is the point spread function of the system,
  - `var_n` is the variance of the noise (and we assume white noise)
- The function have to return the restored image (`x_tilde`).

Exercise description continues on the next slide…

Exercise 4 – continued

First, move everything to the frequency domain (just as in Exercise 2).

Moreover, you have to compute the noise to signal power ratio. In this, we know the variance of the noise (`var_n` is given as an input parameter), and the variance of the signal can be approximated with the variance of the degraded image (`var(y(:))`).

$$NSPR = \frac{\sigma_N^2}{\sigma_X^2}$$

Exercise description continues on the next slide...

Exercise 4 – continued

After this, implement the formula for the `R` restoration matrix:

$$R(\omega_1,\omega_2) = \frac{H^*(\omega_1,\omega_2)}{|H(\omega_1,\omega_2)|^2 + \frac{P_{NN}(\omega_1,\omega_2)}{P_{XX}(\omega_1,\omega_2)}} = R(\omega_1,\omega_2) = \frac{H^*(\omega_1,\omega_2)}{|H(\omega_1,\omega_2)|^2 + NSPR}$$

assuming white noise

in which H^* means complex conjugate (use the `conj()` function in MATLAB).

Finally, apply the restoration system to the input in the frequency domain. Multiply `Y` with `R` elementwise and transform back the result to the spatial domain.

$$\tilde{x} = \left|\mathrm{IDFT}\{R \cdot \mathrm{DFT}\{y\}\}\right|$$

Execute `script4.m` and check your results!

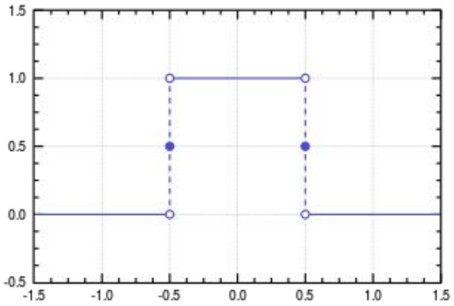
Original input image

Degraded image (filter + noise)

Restored image (Wiener, NSPR)

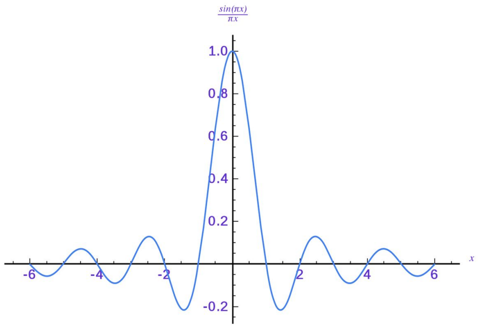
Ringing artifacts

- appears as spurious edges near sharp transitions in the signal
- informally: the main cause is the missing high frequency
- central example: the ideal low pass filter (*sinc filter*):
  - its desired frequency response is the rectangular function:



$$H(f) = \mathrm{rect}\left(\frac{f}{2B}\right) = \begin{cases} 0, & \text{if } |f| > B, \\ \frac{1}{2}, & \text{if } |f| = B, \\ 1, & \text{if } |f| < B, \end{cases}$$

  - its impulse response:

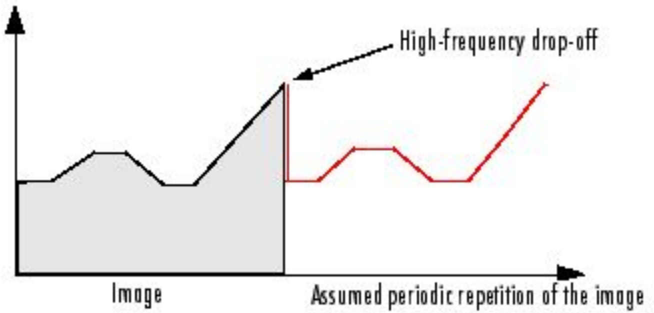


$$h(t) = \mathcal{F}^{-1}\{H(f)\} = \int_{-B}^{B} \exp(2\pi i f t)\, df$$
$$= 2B\,\mathrm{sinc}(2Bt)$$

sources: https://en.wikipedia.org/wiki/Ringing_artifacts and https://en.wikipedia.org/wiki/Sinc_filter

Ringing in deblurred images

- DFT, used by deblurring functions, assumes your image is a periodic signal (as you would repeatedly use it for tiling the 2D floor)
- this assumption creates high-frequency drop-off at the edges of the images



- this high-frequency drop-off creates the effect called *boundary related ringing* in deblurred images

source: https://www.mathworks.com/help/images/avoid-ringing-in-deblurred-images.html

Exercise 5

Implement the function `blur_edge` in which:

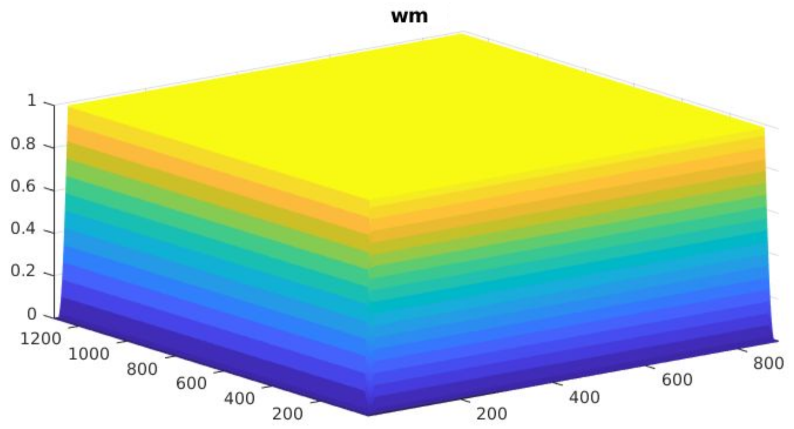
- The function has one input: `y` which is the already distorted image.
- The same image with blurred boundary should be returned (`y_edgetapered`).

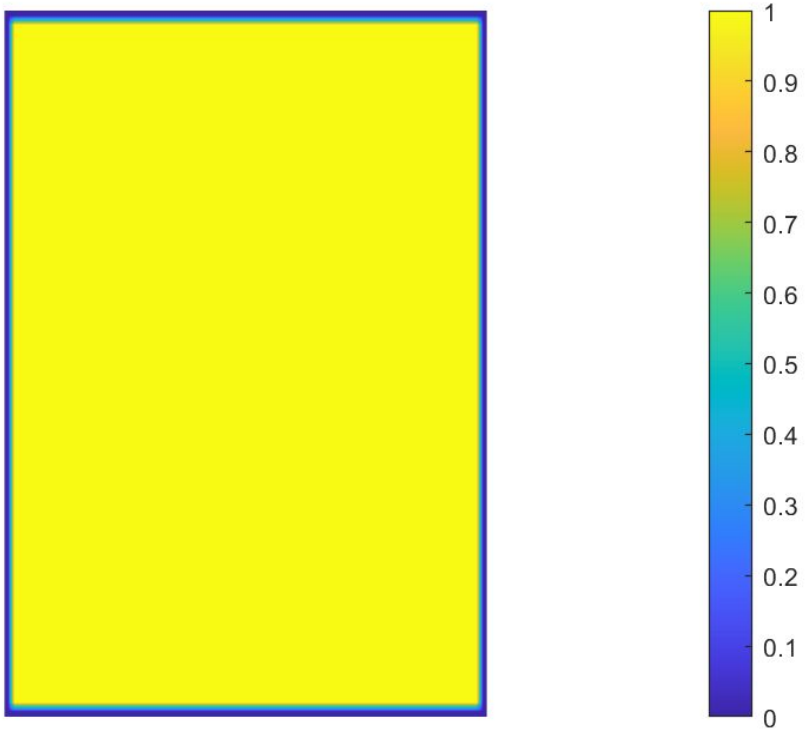
Exercise description continues on the next slide…

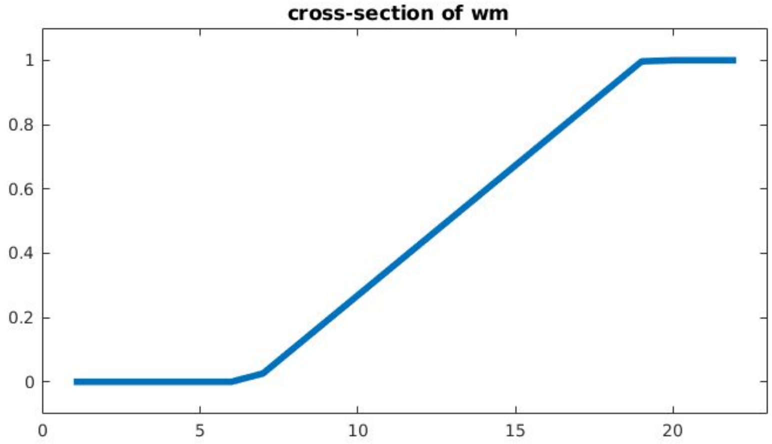
Exercise 5 – continued

- create a *point spread function* with `fspecial`, it should be `'gaussian'`, with *hsize* `60` and *sigma* `10`
- convert it to *optical transfer function* (`psf2otf`), the size should match the size of `y`
- calculate the blurred version of the input with this optical transfer function:
  `blurred_img = | IDFT{OTF · DFT{y} } |`
- create a *weighting matrix* with which you can combine the blurred image with the original one (at the center preserve the original; at the boundaries use the blurred one):
  - create a *70x70* sized array with full of ones (`ones`, let's call it `wm`),
  - pad it with one layer of zeros around itself (`padarray`),
  - resize this array to the original size of the `y` (`imresize`, use the `'bilinear'` option),
- calculate the output:
  `y_edgetapered = wm · y + (1-wm) · blurred_img`

Execute `script5.m` and check your results!







*this figure will not show up, just for illustrative purposes

Degraded image, before edgetaper

Degraded image, after edgetaper

Degraded image (filter + noise)

Wiener (with noise layer), without edgetaper

Wiener (with noise layer), with edgetaper

THE END