Machine Vision Review

Dr. Daniel B. Rowe
Professor of Computational Statistics
Department of Mathematical and Statistical Sciences
Marquette University

Outline

Lecture 01: Matlab
Lecture 02: Within Image Processing
Lecture 03: Image Filter Design
Lecture 04: Statistical Implications
Lecture 05: Pixel Statistics & Template Matching
Lecture 06: Through Image Processing
Lecture 07: The Discrete Fourier Transform
Lecture 08: Convolution via the DFT
Lecture 09: Fast Object Tracking
Lecture 10: Peaks, Valleys, and Ridges

Discussion

Introduction to Matlab

Dr. Daniel B. Rowe
Professor of Computational Statistics
Department of Mathematical and Statistical Sciences
Marquette University

Using Matlab

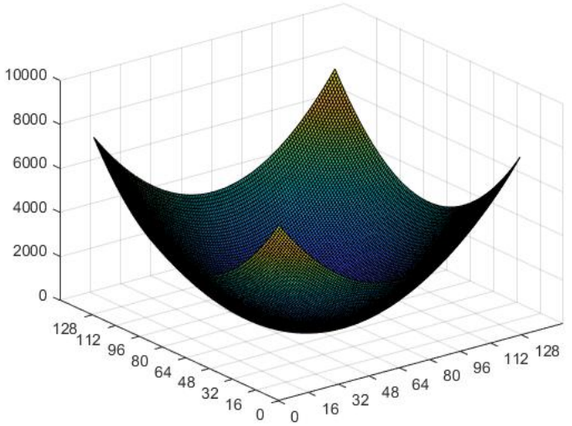
Some students were using Matlab for the first time.

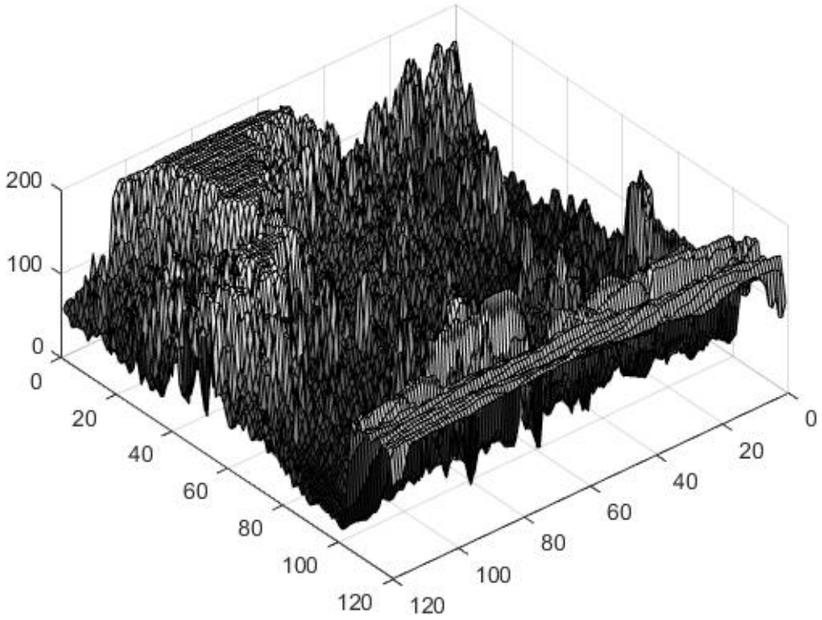
Installing Matlab

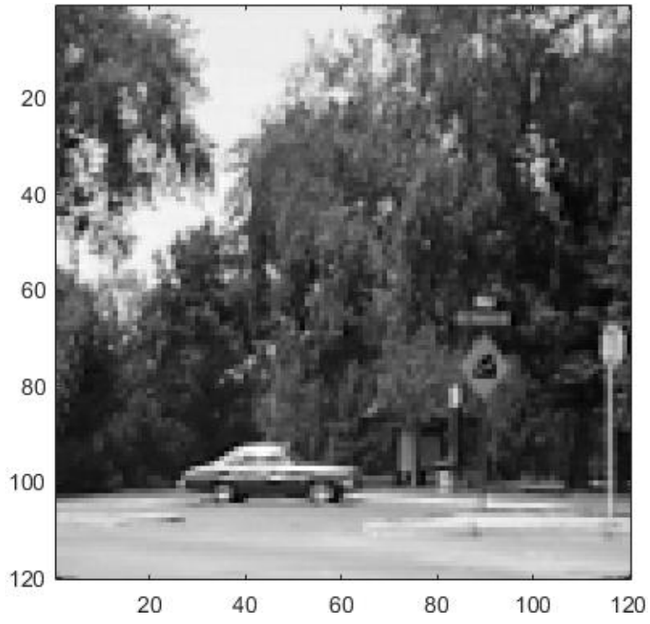
Using Matlab

Saving/Loading from/into Matlab

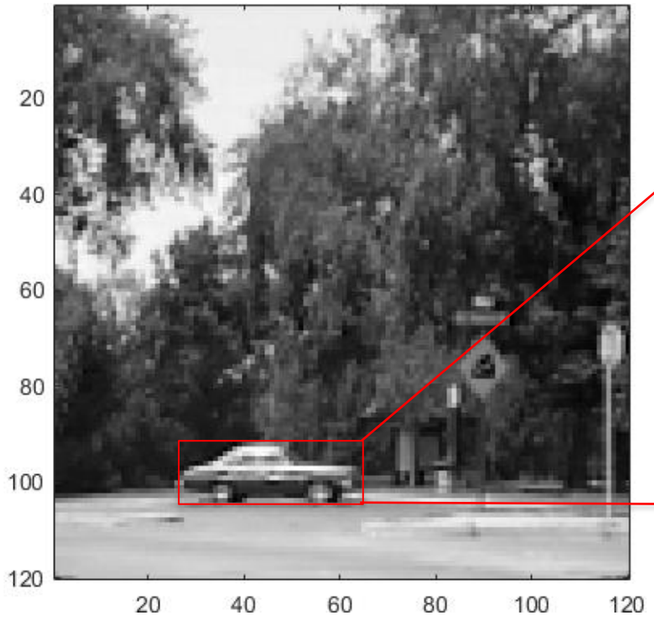
Functions in Matlab

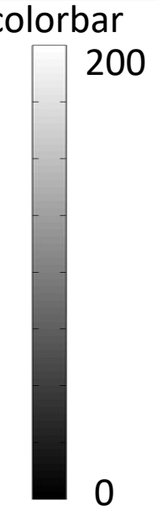


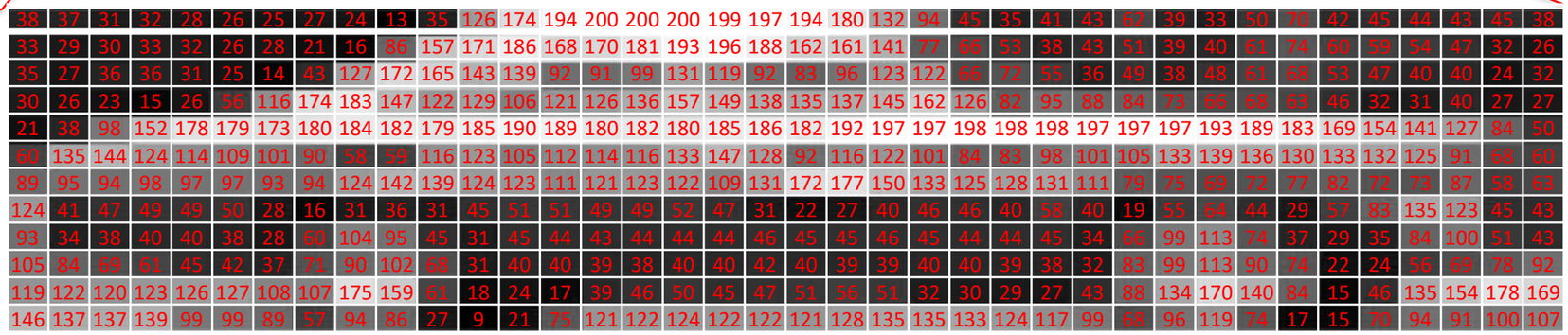




Using Matlab








```
car=fxy(93:104,27:64);
figure;
imagesc(car)
axis image, colormap(gray), axis off
print(gcf,'-dtiffn','-r100',['car'])

filename = 'cardata.xlsx';
writematrix(car,filename,'Sheet',1,'Range','A1')
```

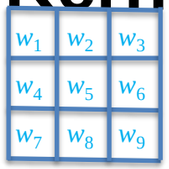
← Save into an excel spreadsheet!

Within Image Processing

Dr. Daniel B. Rowe
Professor of Computational Statistics
Department of Mathematical and Statistical Sciences
Marquette University

Convolution Kernel

Weights w



3x3

$$\sum w_i = 1$$

There are different sizes and weightings to perform different functions.

w_1	w_2	w_3	w_4	w_5
w_6	w_7	w_8	w_9	w_{10}
w_{11}	w_{12}	w_{13}	w_{14}	w_{15}
w_{16}	w_{17}	w_{18}	w_{19}	w_{20}
w_{21}	w_{22}	w_{23}	w_{24}	w_{25}

weights

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

/25

5x5 average

0	0	0	0	0
0	0	1/5	0	0
0	1/5	1/5	1/5	0
0	0	1/5	0	0
0	0	0	0	0

4 neighbor

0	0	0	0	0
0	1/16	1/8	1/16	0
0	1/8	1/4	1/8	0
0	1/16	1/8	1/16	0
0	0	0	0	0

Binomial

1	20	55	20	1
20	403	1097	403	20
55	1097	2981	1097	55
20	403	1097	403	20
1	20	55	20	1

/9365

Gaussian

Smoothing.

Convolution Kernel

Weights w →

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

3x3

$$\sum w_i = 0$$

There are different sizes and weightings to perform different functions.

w_1	w_2	w_3	w_4	w_5
w_6	w_7	w_8	w_9	w_{10}
w_{11}	w_{12}	w_{13}	w_{14}	w_{15}
w_{16}	w_{17}	w_{18}	w_{19}	w_{20}
w_{21}	w_{22}	w_{23}	w_{24}	w_{25}

weights

0	0	0	0	0
0	-1	-1	-1	0
0	0	0	0	0
0	1	1	1	0
0	0	0	0	0

Gradient

0	0	0	0	0
0	-1	0	1	0
0	-2	0	2	0
0	-1	0	1	0
0	0	0	0	0

Sobel

0	0	0	0	0
0	0	1	0	0
0	1	-4	1	0
0	0	1	0	0
0	0	0	0	0

Laplacian

0	0	0	0	0
0	0	-1	-1	0
0	1	0	-1	0
0	1	1	0	0
0	0	0	0	0

Oblique Gradient

Sharpening.

Image Smoothing

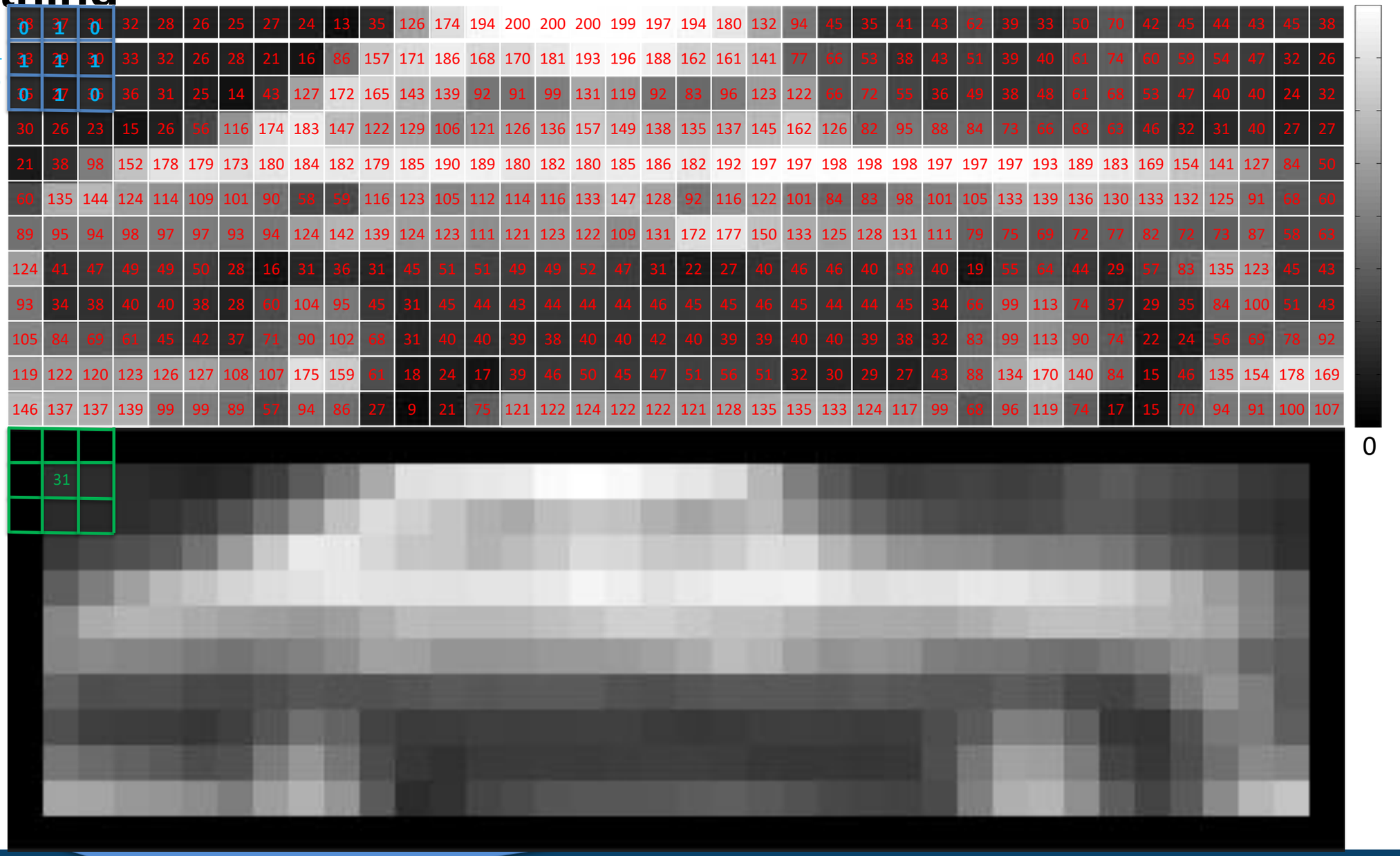
Weights

w

$\frac{1}{5}$

Pixel Values

p



New image

Within Image Processing

0	1/5	0
1/5	1/5	1/5
0	1/5	0

4 neighbor

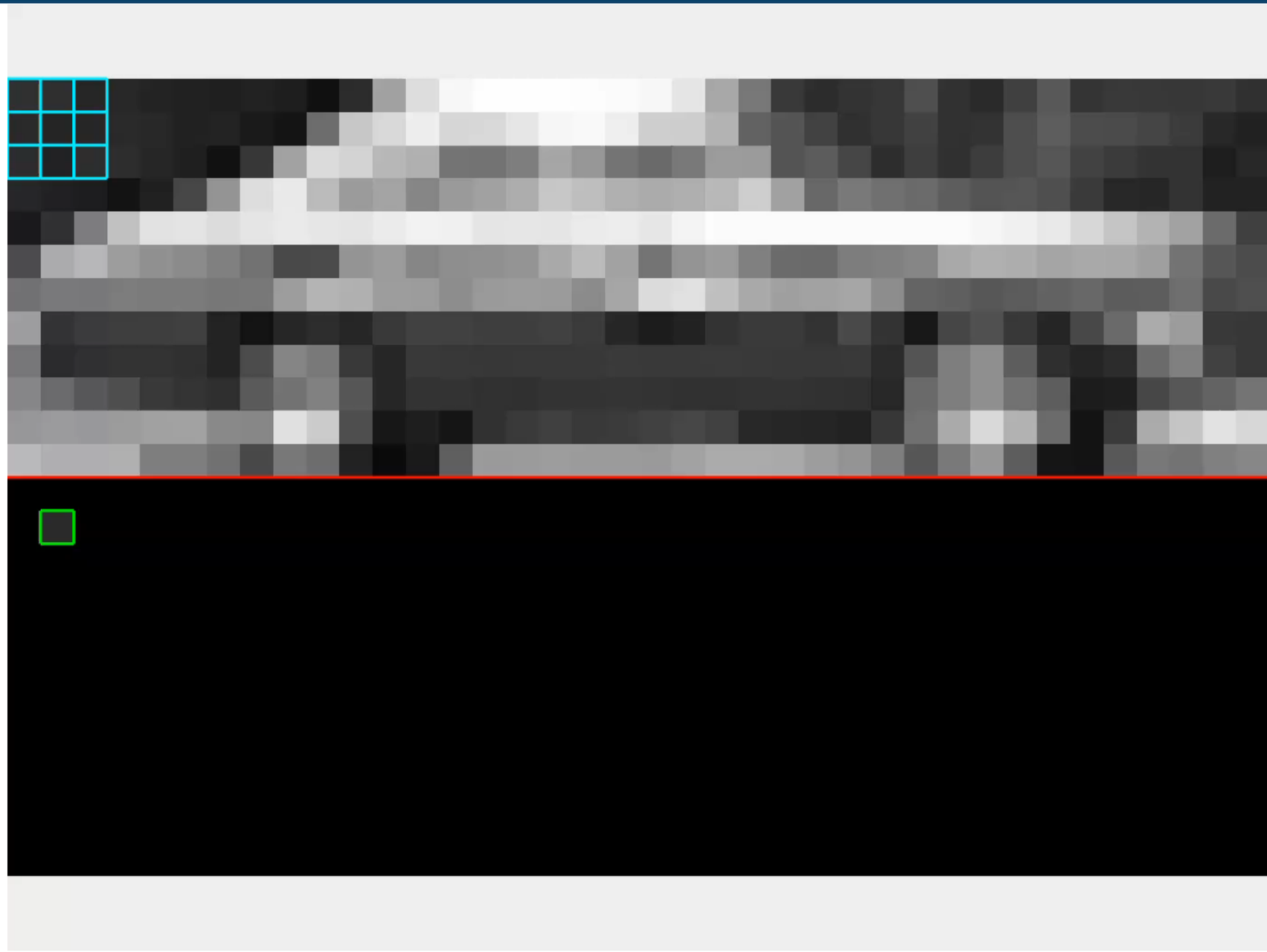


Image Filter Design

Dr. Daniel B. Rowe
Professor of Computational Statistics
Department of Mathematical and Statistical Sciences
Marquette University

Smoothing Filters

The zero mean Gaussian distribution with common variance σ^2 is

$$g(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}.$$ (Note that we have neglected the normalizing constant.)

We can form a 5×5 Gaussian filter with $\sigma^2=0.5$.

Calculate the unnormalized weights.

Divide all the values by the corners value.

Round to the nearest integer.

Divide by the sum of the integers.

0.000335462627903

.0003	.0067	.0183	.0067	.0003
.0067	.1353	.3679	.1353	.0067
.0183	.3679	1.0000	.3679	.0183
.0067	.1353	.3679	.1353	.0067
.0003	.0067	.0183	.0067	.0003

1	20	55	20	1
20	403	1097	403	20
55	1097	2981	1097	55
20	403	1097	403	20
1	20	55	20	1

/9365

Image Smoothing

Gaussian

1	20	55	20	1
20	403	1097	403	20
55	1097	2981	1097	55
20	403	1097	403	20
1	20	55	20	1

/9365

Apply to whole image and examine the difference

Original

Smoothed

Difference

Sharpening Filters

The discrete version of the first derivatives $\frac{\partial}{\partial x} f(x,y)$ and $\frac{\partial}{\partial y} f(x,y)$

are $D_x = f(x,y) - f(x-1,y)$ and $D_y = f(x,y) - f(x,y-1)$ which in terms of kernels are

-1	1

and

-1
1

but may also be expressed as

-1	1
-1	1

and

-1	-1
1	1

or even with larger kernels.

Prewitt's 3×3 derivative kernels are

-1	-1	-1
0	0	0
1	1	1

and

-1	0	1
-1	0	1
-1	0	1

.

Derivative at center pixel.

Roberts cross gradient at 45° are

-1	0
0	1

and

0	-1
1	0

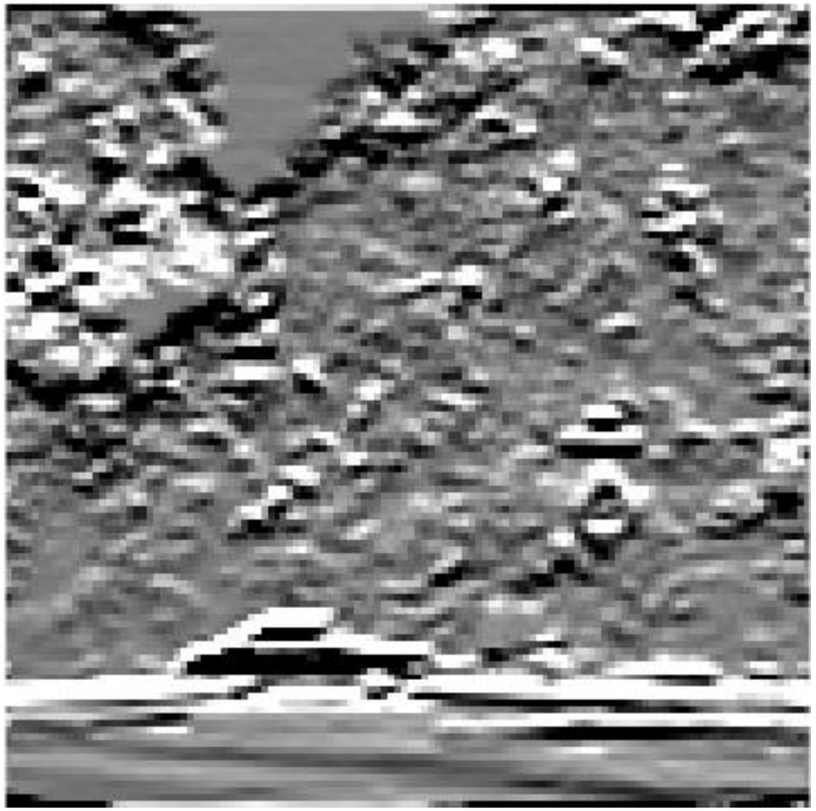
.

Sharpening Filters

Derivative at center pixel.

-1	-1	-1
0	0	0
1	1	1

Original



U-D Gradient

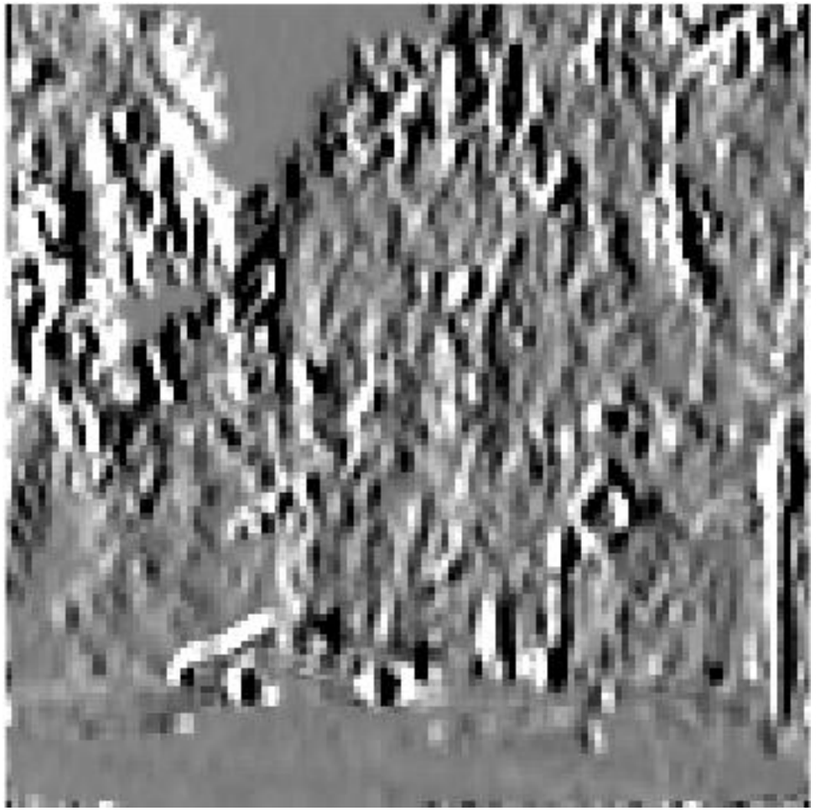
Difference

Sharpening Filters

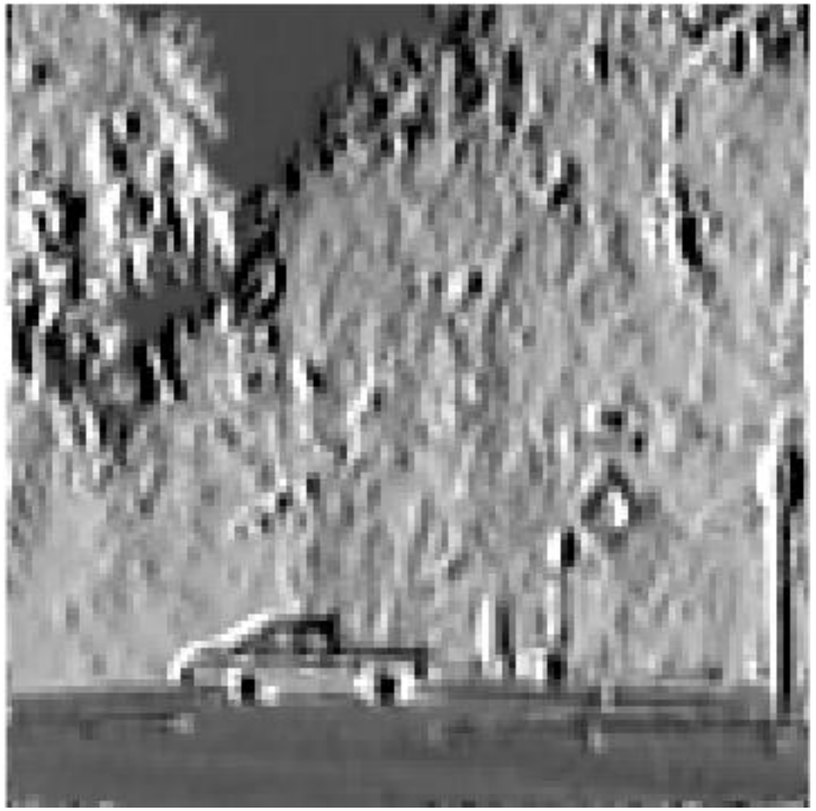
Derivative at center pixel.

-1	0	1
-1	0	1
-1	0	1

Original



L-R Gradient



Difference

200,
100,100

0,
-100,-300

Hybrid Filters

We can boost the edges in an image with a “high boost” filter.

Original

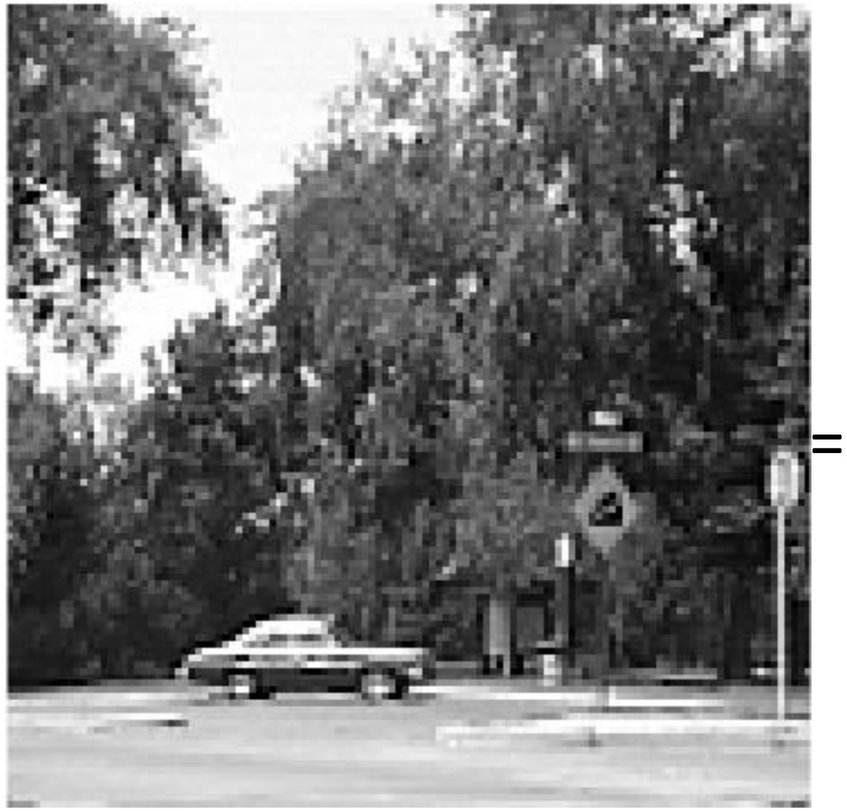
Smoothed

High

This can be thought of as subtracting the low fom the original image.

Hybrid Filters

We add back part of the high to the original to boost the high.



=2

-

High Boost

Original

Low

HighBoost=A*(Original)-(Low)

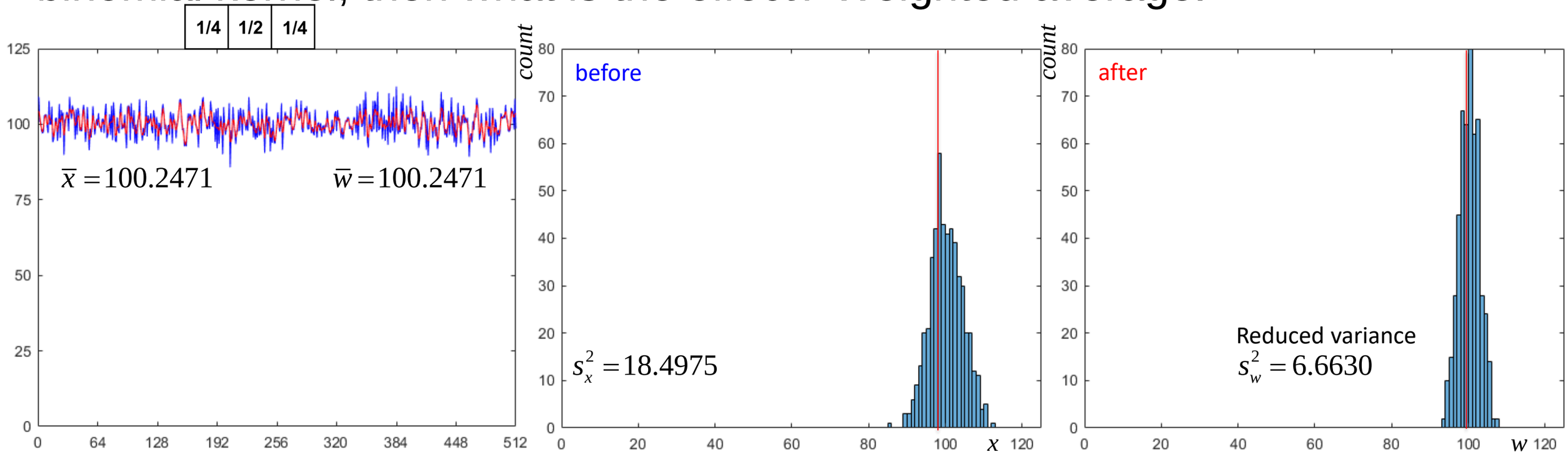
HighBoost =255(HighBoost) /max(max(HighBoost)); %renormalize

Statistical Implications

Dr. Daniel B. Rowe
Professor of Computational Statistics
Department of Mathematical and Statistical Sciences
Marquette University

Introduction

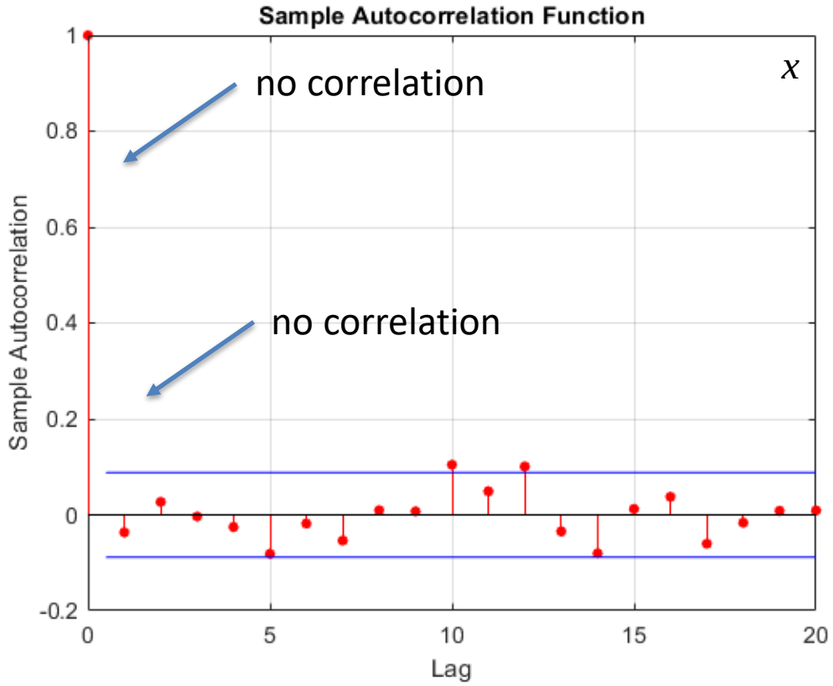
If we were to smooth the time series (row of pixels) with say a 3 point binomial kernel, then what is the effect? Weighted average.

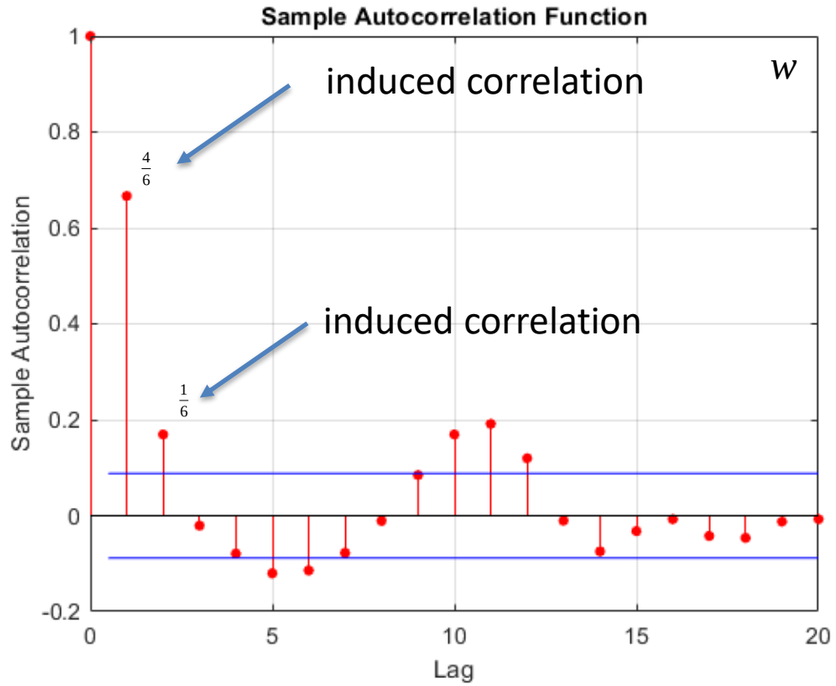


Reduced variance but induced temporal correlation.

Introduction

We can calculate the temporal autocorrelation and see induced correlation.





Make a copy of time series shifted by lag L with wrap around, then calculate The correlation between the two time series.

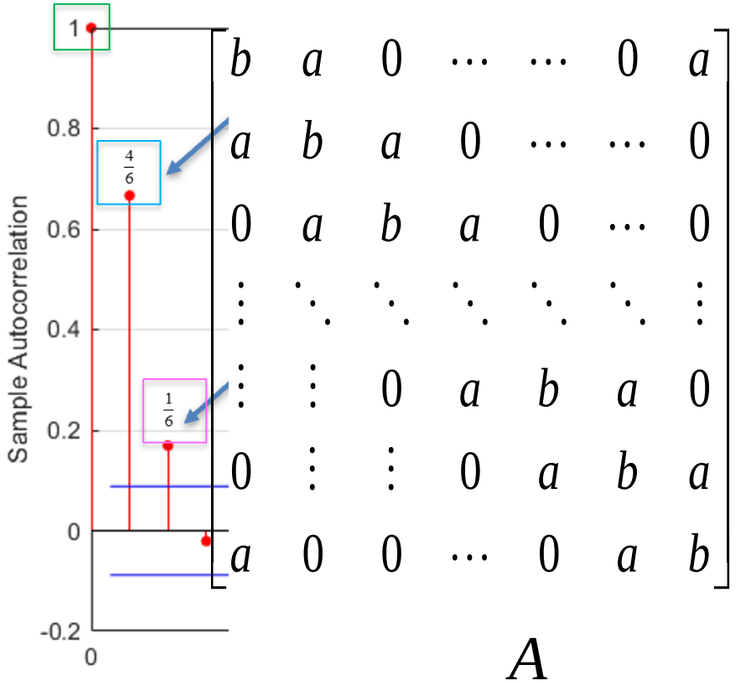
L	$corx$	$corw$	
0	1.0000	1.0000	
1	-0.0382	**0.6707**	$\frac{4}{6}$
2	0.0344	**0.1757**	$\frac{1}{6}$
3	-0.0049	-0.0162	
4	-0.0210	-0.0768	
5	-0.0791	-0.1179	
6	-0.0147	-0.1125	
7	-0.0577	-0.0785	
8	0.0072	-0.0151	
9	0.0022	0.0800	
10	0.1061	0.1681	
11	0.0519	0.1997	
12	0.1088	0.1384	
13	-0.0220	0.0073	
14	-0.0805	-0.0661	
15	0.0098	-0.0304	
16	0.0428	-0.0082	
17	-0.0609	-0.0446	
18	-0.0187	-0.0516	
19	0.0025	-0.0211	
20	0.0056	-0.0209	

Let's examine how and why does this happen?
This also happens in a 2D image!

Time Series Statistics

This slide assumes $\sigma^2=1$.

Because there is a two step correlation between pixels in the time series (row), we can determine the weights in the kernel used for convolution.



$$\begin{bmatrix} b & a & 0 & \cdots & \cdots & 0 & a \\ a & b & a & 0 & \cdots & \cdots & 0 \\ 0 & a & b & a & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & 0 & a & b & a & 0 \\ 0 & \vdots & \vdots & 0 & a & b & a \\ a & 0 & 0 & \cdots & 0 & a & b \end{bmatrix} \times \begin{bmatrix} b & a & 0 & \cdots & \cdots & 0 & a \\ a & b & a & 0 & \cdots & \cdots & 0 \\ 0 & a & b & a & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & 0 & a & b & a & 0 \\ 0 & \vdots & \vdots & 0 & a & b & a \\ a & 0 & 0 & \cdots & 0 & a & b \end{bmatrix} = \begin{bmatrix} b^2+2a^2 & 2ab & a^2 & 0 & \cdots & 0 & a^2 \\ 2ab & b^2+2a^2 & 2ab & a^2 & \cdots & \cdots & 0 \\ a^2 & 2ab & b^2+2a^2 & 2ab & a^2 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & a^2 & 2ab & b^2+2a^2 & 2ab & a^2 \\ a^2 & \vdots & \vdots & a^2 & 2ab & b^2+2a^2 & 2ab \\ 2ab & a^2 & 0 & \cdots & a^2 & 2ab & b^2+2a^2 \end{bmatrix}$$

A $\qquad$ A' $\qquad$ $AA'=cor(w)$

$b^2+2a^2=1$ $\qquad$ $2ab=\frac{2}{3}$ $\qquad$ $a^2=\frac{1}{6}$ $\longrightarrow$ $a=\frac{1}{\sqrt{6}}, b=\frac{\sqrt{6}}{3}$ mult by $\frac{\sqrt{6}}{4}$ for $a=\frac{1}{4}, b=\frac{1}{2}$.

Form $B=A^{-1}$ and deconvolve!

Time Series Statistics

Unsmoothing a time series with a deconvolution filter can be written as a matrix multiplication.

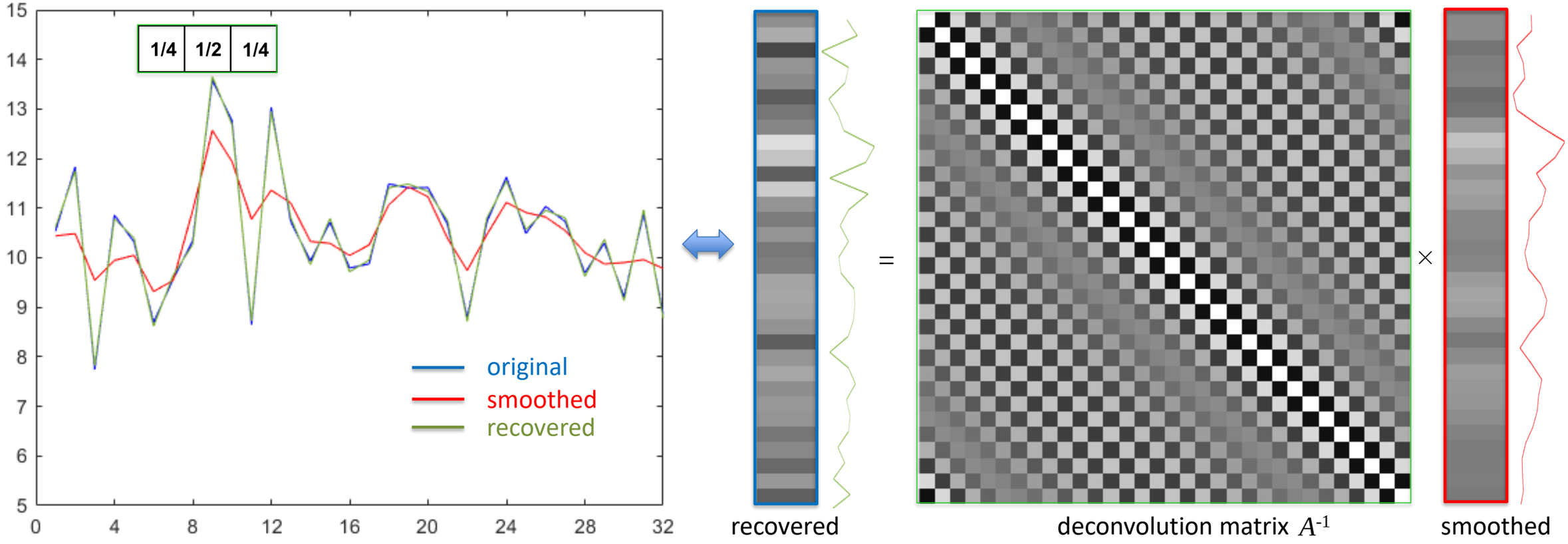


Image Statistics

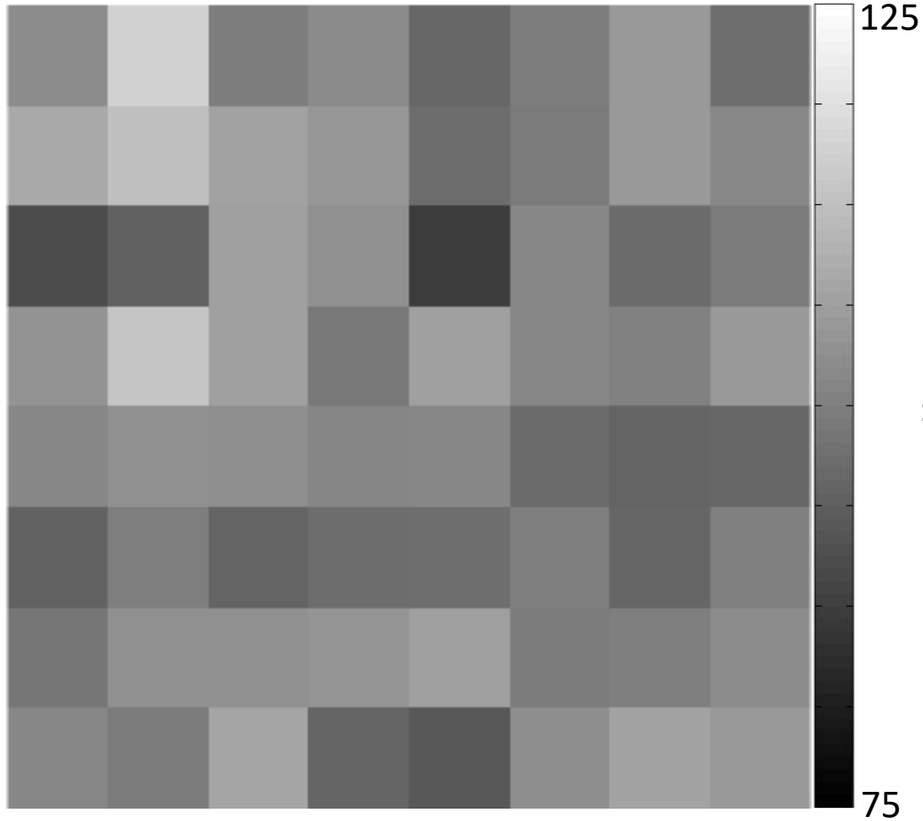
Imagine we have the below 8×8 noisy image and smooth with kernel.

0	1/8	0
1/8	1/2	1/8
0	1/8	0

Kernel

$*$

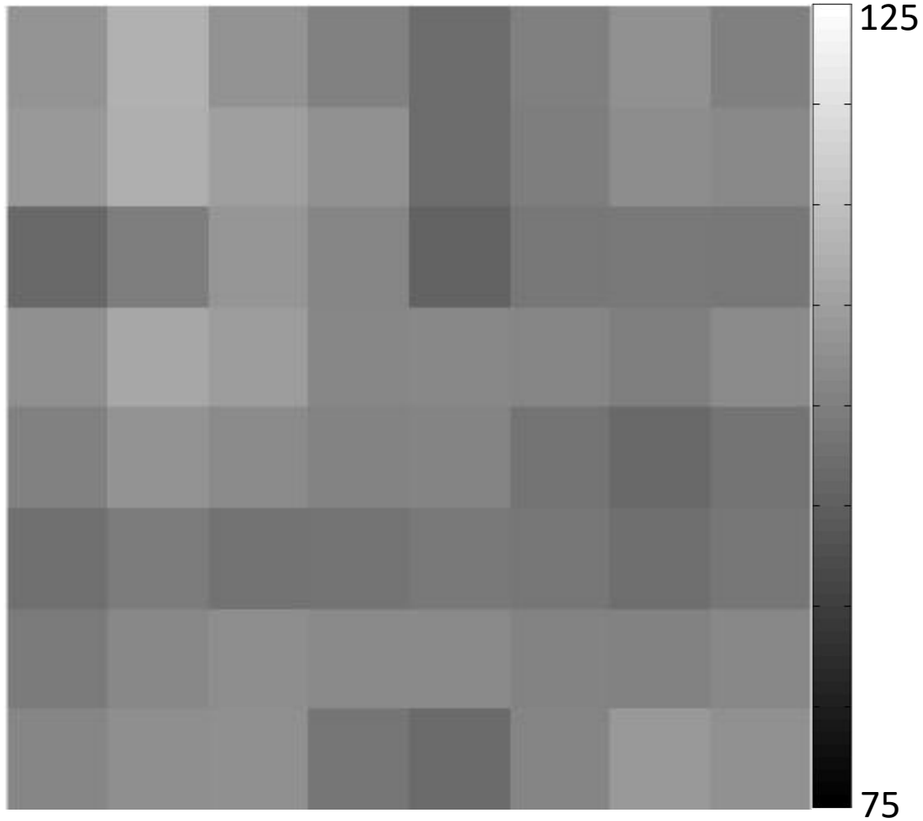
convolution



Input Image

$\bar{I} = 100.9576 \qquad s_I^2 = 28.4930$

$=$



Output Image

$\bar{O} = 100.9576 \qquad s_O^2 = 10.4009$

Image Statistics

0	1/8	0
1/8	1/2	1/8
0	1/8	0

Then perform convolution via a convolution matrix.

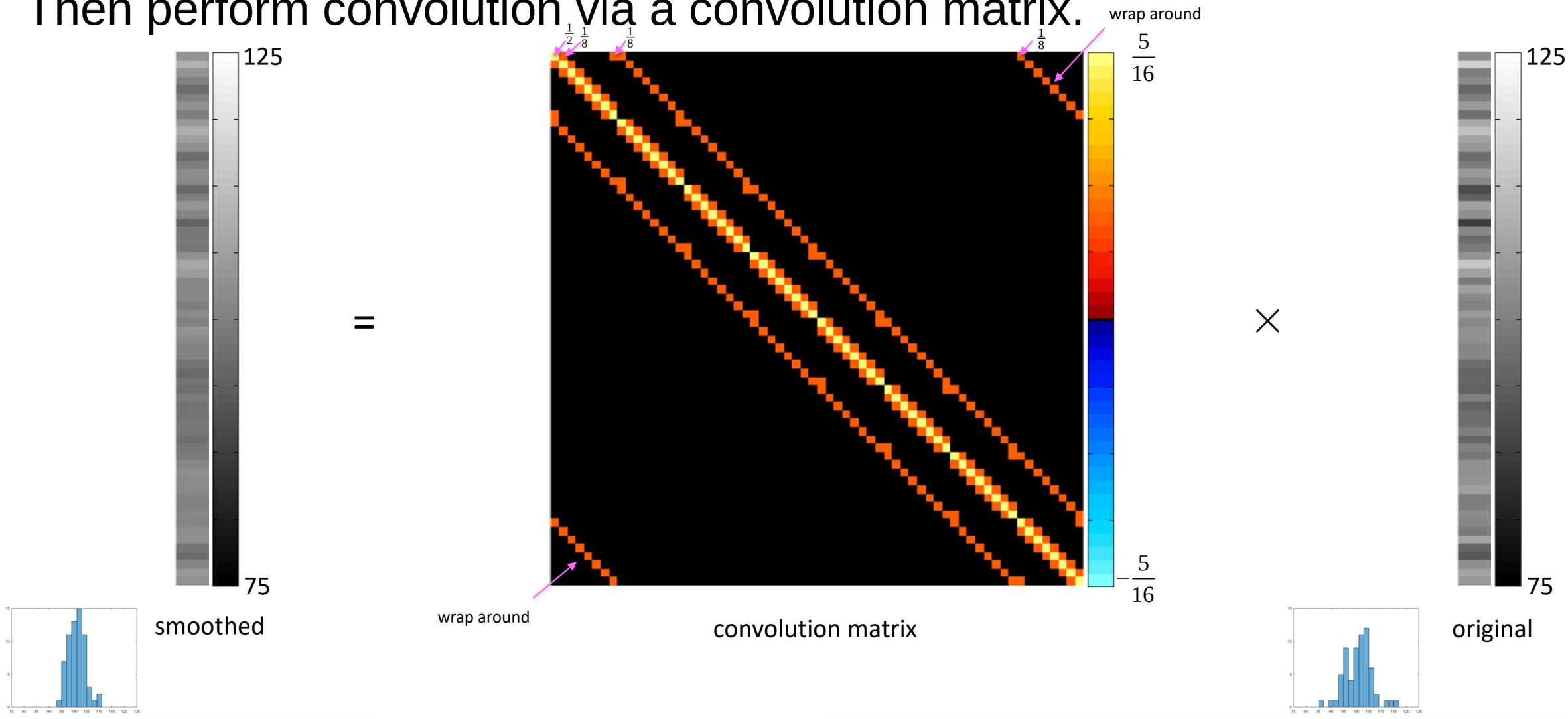
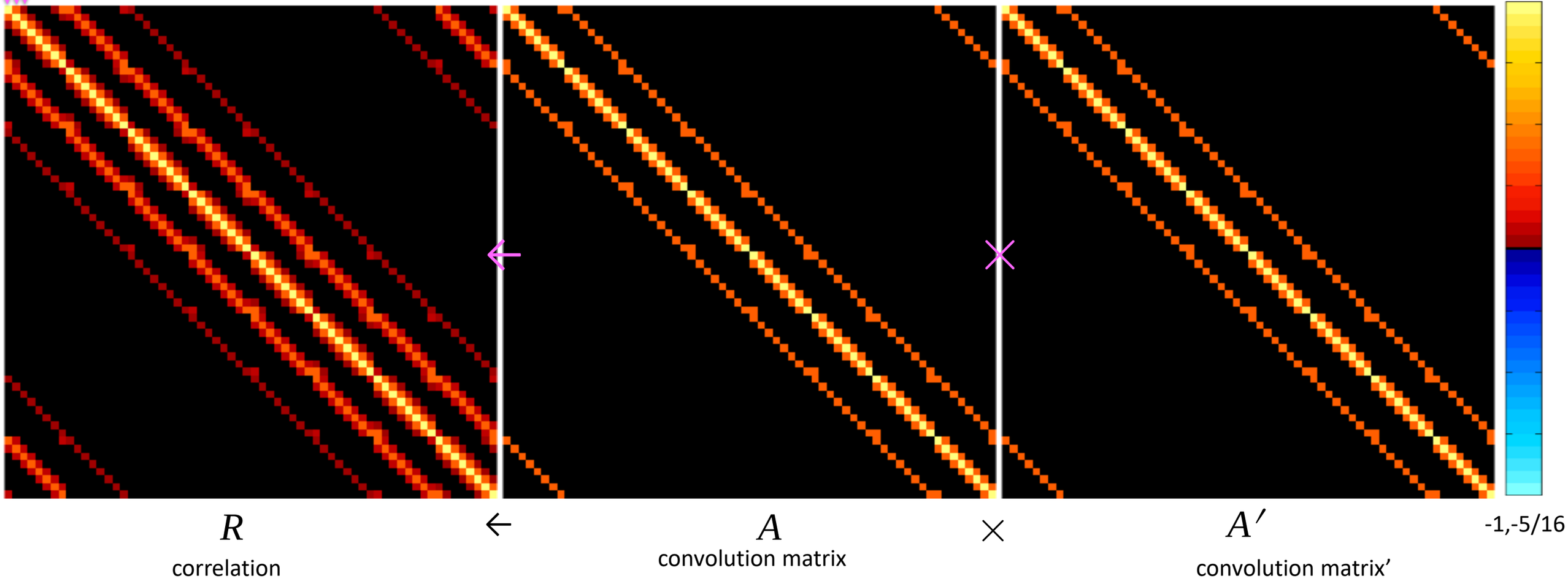


Image Statistics

0	1/8	0
1/8	1/2	1/8
0	1/8	0

We can calculate theoretically what the induced correlation is.



R correlation

$\leftarrow$

A convolution matrix

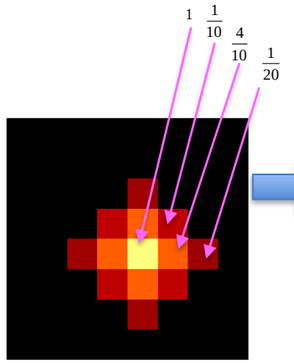
$\times$

A' convolution matrix'

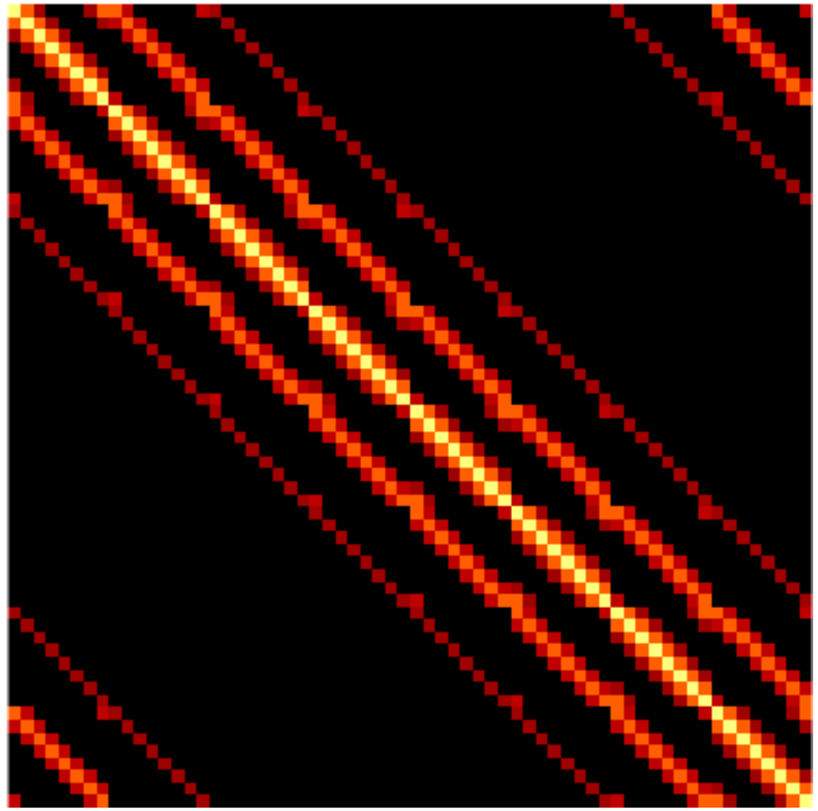
Image Statistics

0	1/8	0
1/8	1/2	1/8
0	1/8	0

If we had the induced correlation, we could calculate convolution matrix.

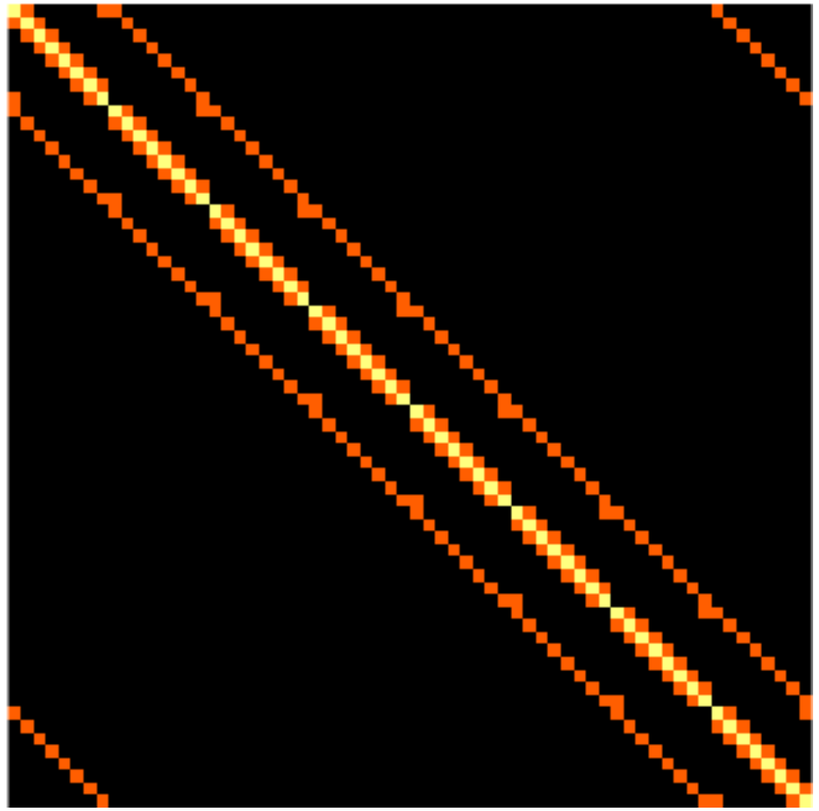


Correlation between (5,5) pixel and all others. Reshaped 37th row of R.



R

correlation



A

convolution matrix

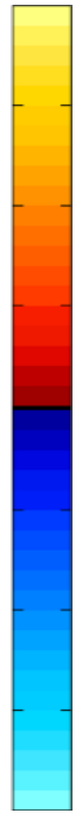
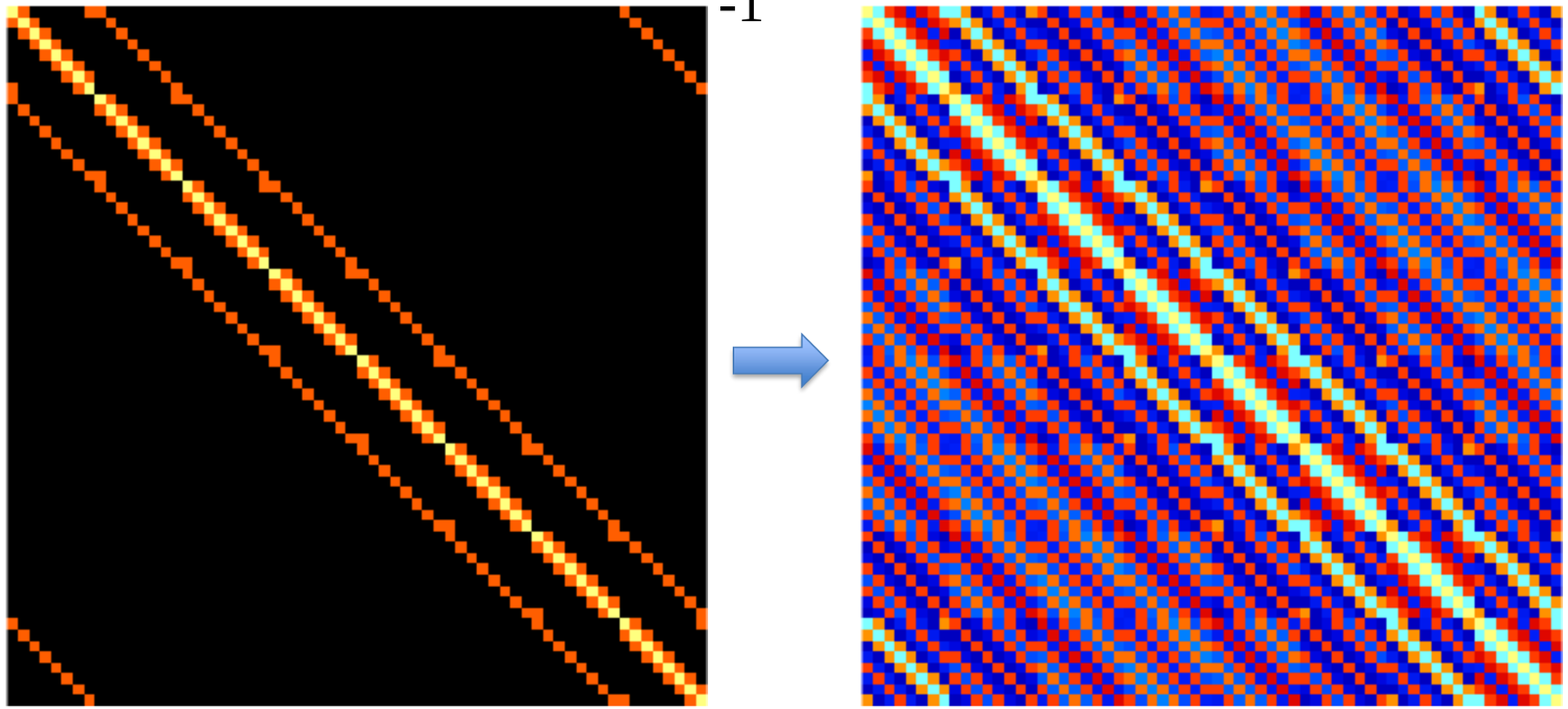


Image Statistics

0	1/8	0
1/8	1/2	1/8
0	1/8	0

Calculate inverse of convolution matrix for a deconvolution matrix.



A

convolution matrix

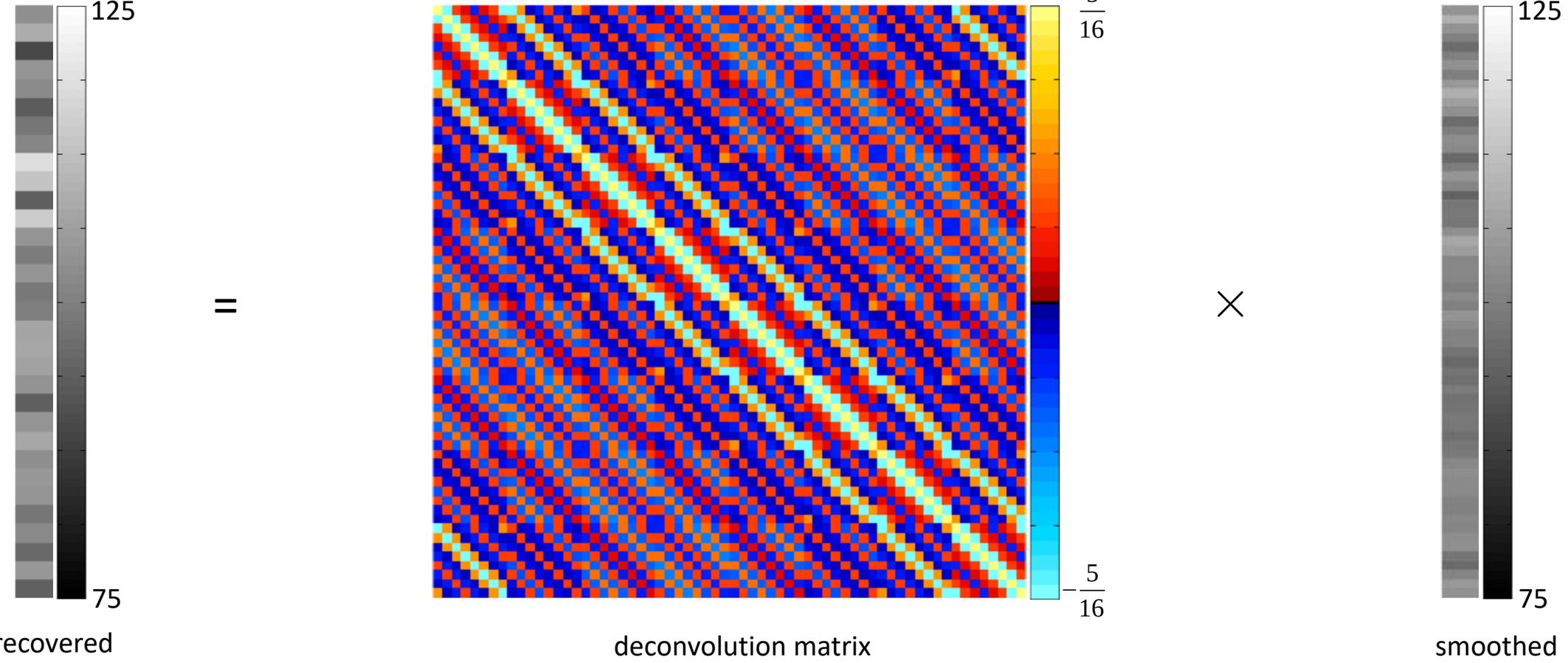
A^{-1}

deconvolution matrix

Image Statistics

0	1/8	0
1/8	1/2	1/8
0	1/8	0

Then perform deconvolution via a deconvolution matrix.



The Correlation Coefficient

Dr. Daniel B. Rowe
Professor of Computational Statistics
Department of Mathematical and Statistical Sciences
Marquette University

The Correlation Distribution

From the variances s_1^2 , s_2^2 , and covariance s_{12} we can perform a transformation of variable to obtain the correlation coefficient $r = \frac{s_{12}}{s_1 s_2}$.

It has been shown that the alternative hypothesis ($\rho \neq 0$) PDF is

$$f(r) = \frac{n-2}{\sqrt{2\pi}} \frac{\Gamma(n-1)}{\Gamma\left(n-\frac{1}{2}\right)} \frac{(1-\rho^2)^{\frac{n-1}{2}} (1-r^2)^{\frac{n-4}{2}}}{(1-\rho r)^{n-\frac{3}{2}}} \; {}_2F_1\left(\tfrac{1}{2}, \tfrac{1}{2}, n-\tfrac{1}{2}, \tfrac{1}{2}(1+\rho r)\right)$$

F is the hypergeometric function

which under the null hypothesis ($\rho = 0$) becomes

$$f(r \mid H_0) = \frac{\Gamma\left(\frac{n-1}{2}\right)}{\pi^{\frac{1}{2}} \Gamma\left(\frac{n-2}{2}\right)} (1-r^2)^{\frac{n-4}{2}}.$$

https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

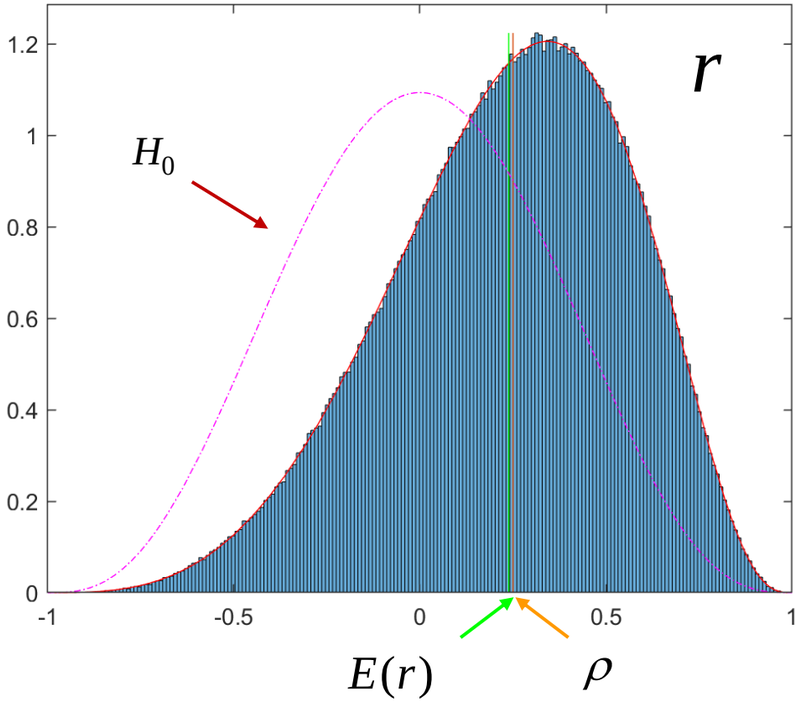
Hotelling: New Light on the Correlation Coefficient and its Transforms. JRSS-B, 15(2), 193-232, 1953.

The Correlation Distribution

Example: Using the same $S_{(1)},\ldots,S_{(L)}$ calculated $r_{(1)},\ldots,r_{(L)}$.

$$\underset{2\times 2}{\Sigma}=\begin{pmatrix}4 & 2\\ 2 & 16\end{pmatrix}$$

$$\rho = 0.25$$



$$E(r) \approx 0.2383$$

$$\bar{r}_{adj} = 0.2453$$

Of note is that $E(r)$ is biased.

$$E(r) = \int_{-1}^{1} r f(r)\, dr$$

$$f(r) = \frac{n-2}{\sqrt{2\pi}} \frac{\Gamma(n-1)}{\Gamma\left(n-\frac{1}{2}\right)} \frac{(1-\rho^2)^{\frac{n-1}{2}}(1-r^2)^{\frac{n-4}{2}}}{(1-\rho r)^{n-\frac{3}{2}}} \; {}_2F_1\left(\tfrac{1}{2},\tfrac{1}{2},n-\tfrac{1}{2},\tfrac{1}{2}(1+\rho r)\right)$$

$$E(r) = \rho + (1-\rho^2)\left(-\frac{\rho}{2n} - \frac{\rho - 9\rho^3}{8n^2} + \frac{\rho + 42\rho^3 - 75\rho^5}{16n^3} + \cdots\right)$$

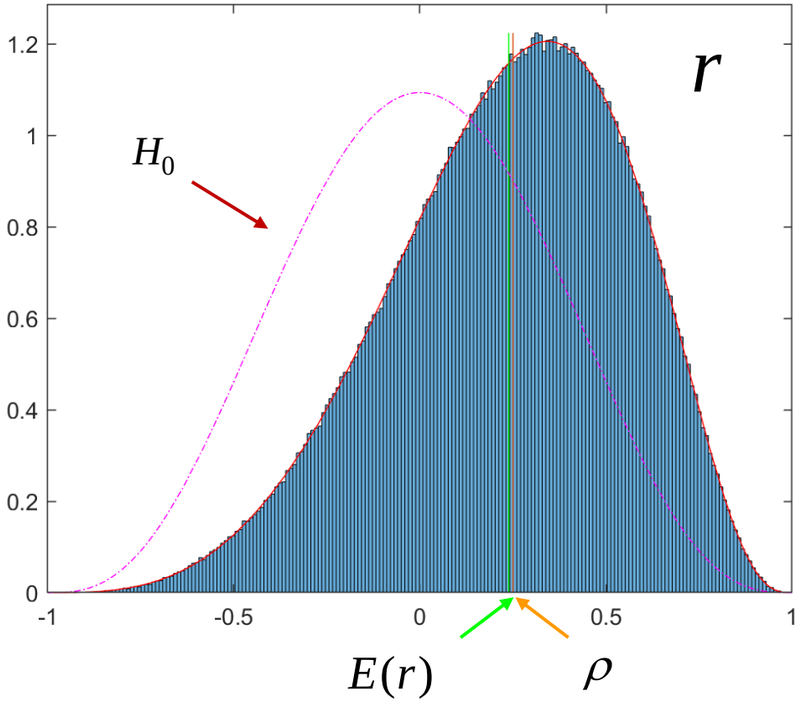
$$E(r) \approx \rho - \frac{\rho(1-\rho^2)}{2n} \quad \rightarrow \quad r_{adj} \approx r\left[1 + \frac{1-r^2}{2n}\right]$$

The Correlation Distribution

Example: Using the same $S_{(1)},\ldots,S_{(L)}$ calculated $r_{(1)},\ldots,r_{(L)}$.

$$\underset{2\times 2}{\Sigma}=\begin{pmatrix}4 & 2\\ 2 & 16\end{pmatrix}$$

$$\rho = 0.25$$



$$E(r) \approx 0.2383$$

$$\bar{r}_{adj} = 0.2453$$

Of note is that $E(r)$ is biased.

$$E(r) = \int_{-1}^{1} r f(r)\, dr$$

$$f(r) = \frac{n-2}{\sqrt{2\pi}} \frac{\Gamma(n-1)}{\Gamma\left(n-\frac{1}{2}\right)} \frac{(1-\rho^2)^{\frac{n-1}{2}}(1-r^2)^{\frac{n-4}{2}}}{(1-\rho r)^{n-\frac{3}{2}}} \; {}_2F_1\left(\tfrac{1}{2},\tfrac{1}{2},n-\tfrac{1}{2},\tfrac{1}{2}(1+\rho r)\right)$$

$$E(r) = \rho + (1-\rho^2)\left(-\frac{\rho}{2n} - \frac{\rho - 9\rho^3}{8n^2} + \frac{\rho + 42\rho^3 - 75\rho^5}{16n^3} + \cdots\right)$$

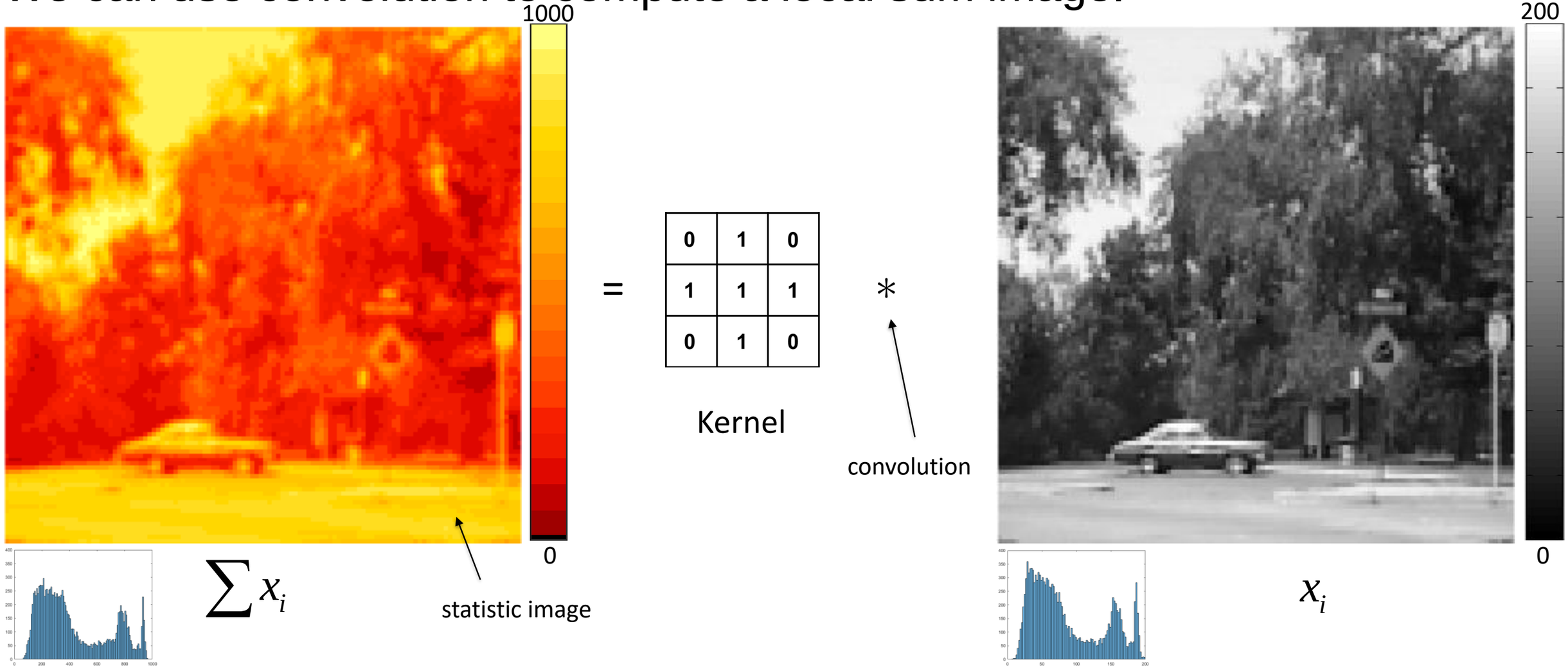
$$E(r) \approx \rho - \frac{\rho(1-\rho^2)}{2n} \quad \rightarrow \quad r_{adj} \approx r\left[1 + \frac{1-r^2}{2n}\right]$$

Pixel Statistics & Template Matching

Dr. Daniel B. Rowe
Professor of Computational Statistics
Department of Mathematical and Statistical Sciences
Marquette University

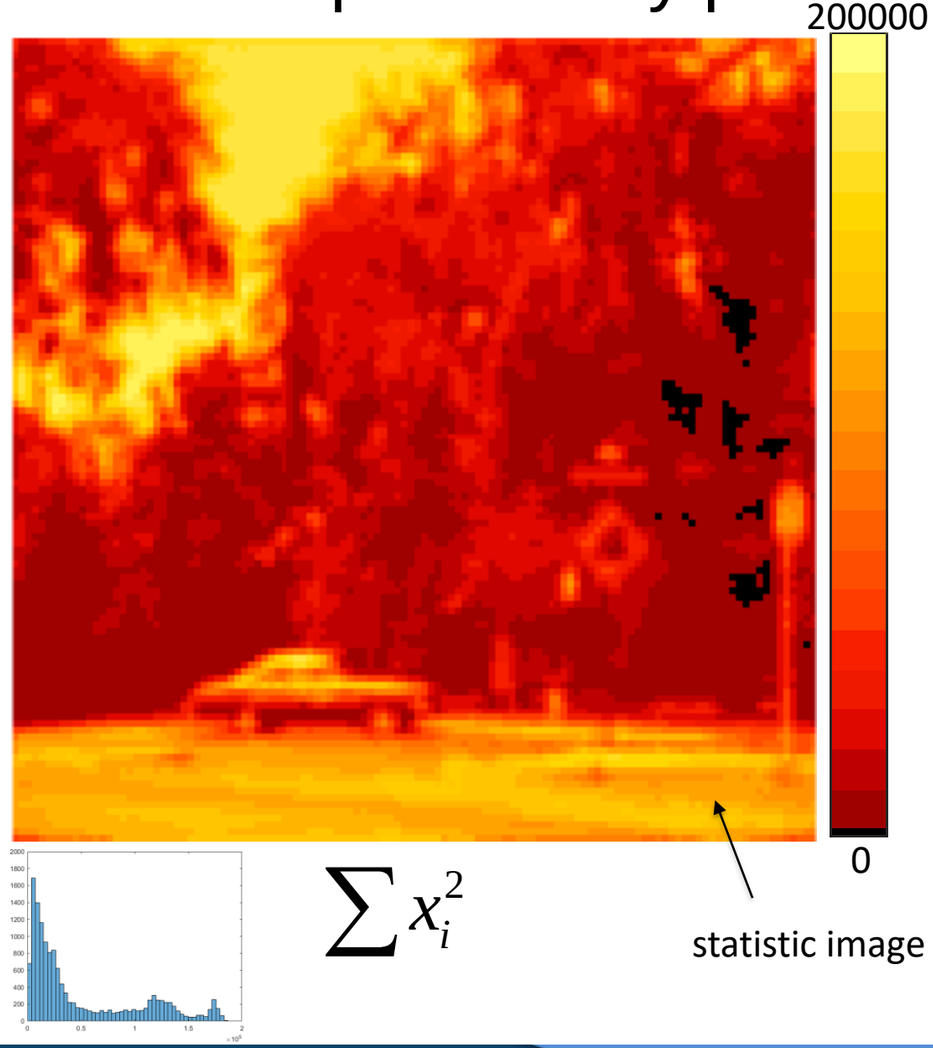
Pixel Statistics

We can use convolution to compute a local sum image.



Pixel Statistics

We can square every pixel value and compute a sum of sqaures.



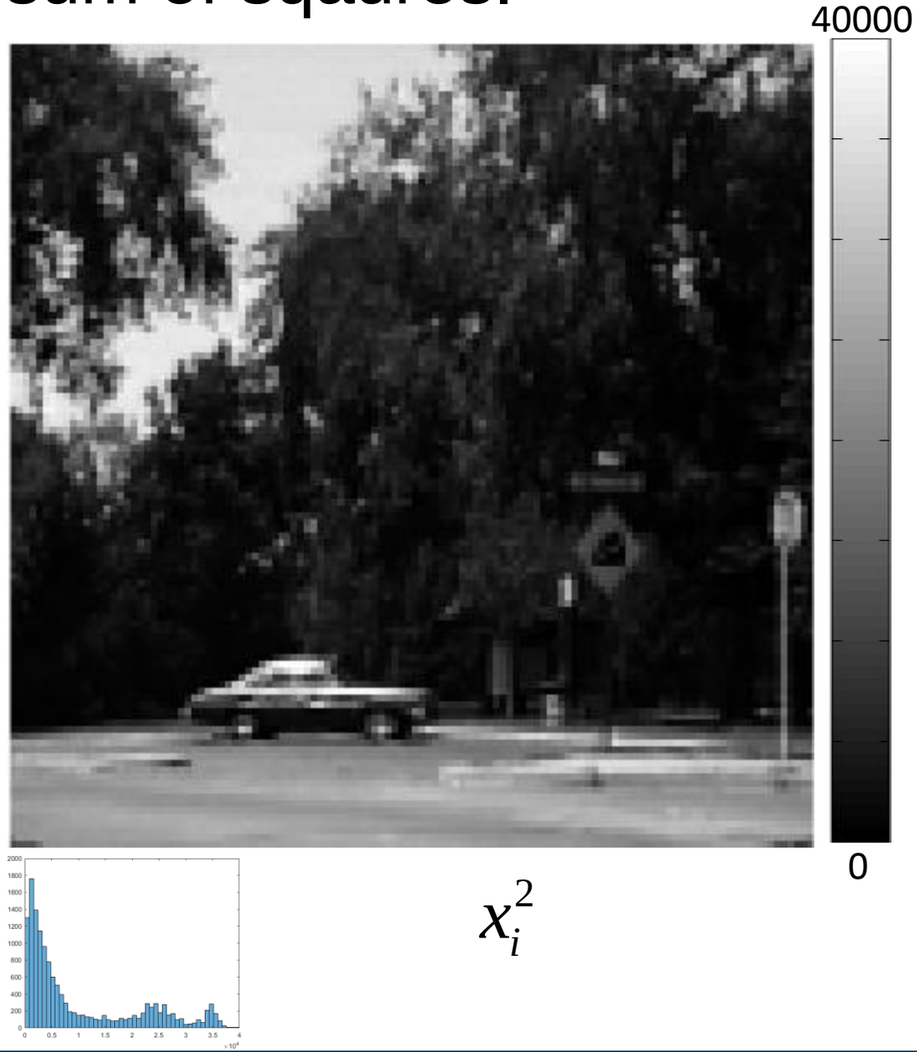
=

0	1	0
1	1	1
0	1	0

Kernel

$*$

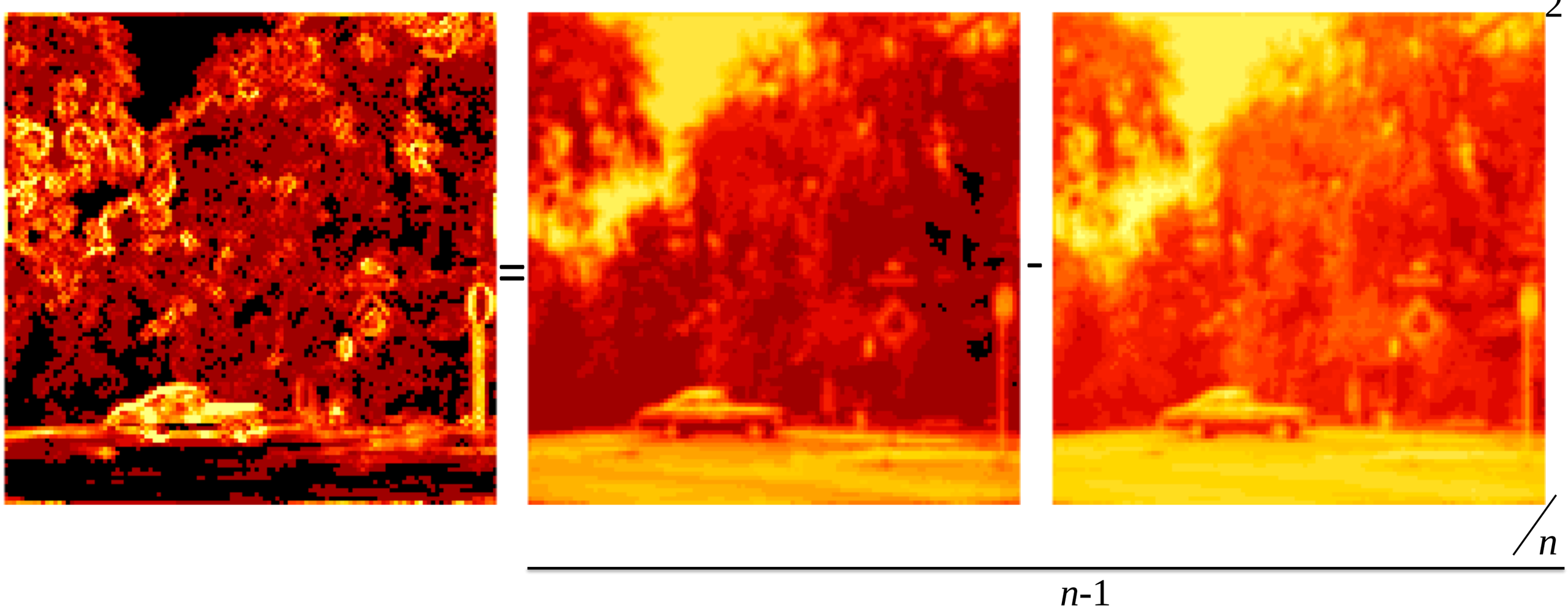
convolution



Pixel Statistics

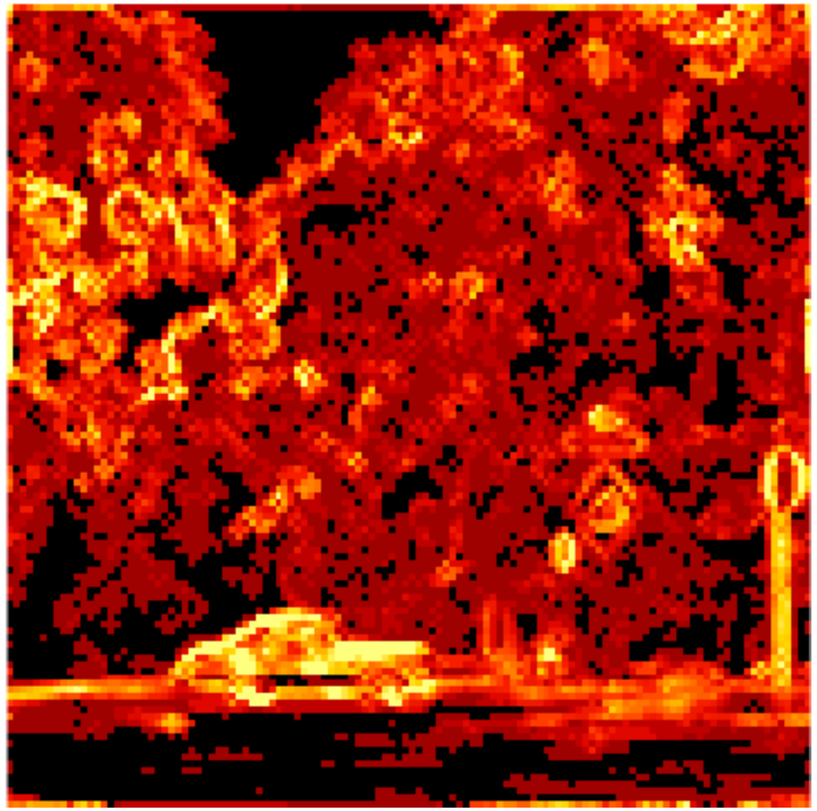
$$s^2 = \frac{\sum x_i^2 - \left(\sum x_i\right)^2 / n}{n-1}$$

We can use sum and sum of squares to compute usual the local variance.



Pixel Statistics

The variance is low in homogeneous and high in heterogeneous areas.



Variance

Original

Mean

Perhaps we could smooth more in low variance areas?

Template Matching

We want to perform computations as fast as possible.

For our object template we can pre-compute

$\sum o_i$ $\sum o_i^2$

Then as we loop through our image, compute

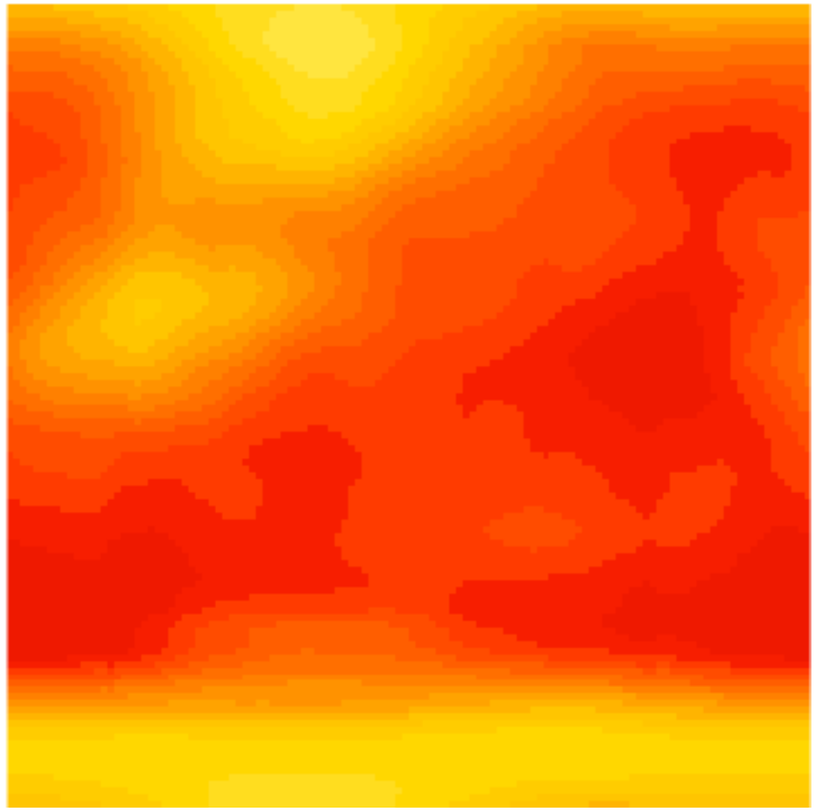
$\sum p_i$ $\sum p_i^2$ $\sum o_i p_i$

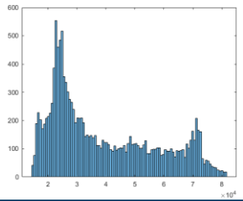
$$r = \frac{\sum p_i o_i - \frac{1}{n}\left(\sum p_i\right)\left(\sum o_i\right)}{\sqrt{\sum p_i^2 - \frac{1}{n}\left(\sum p_i\right)^2}\sqrt{\sum o_i^2 - \frac{1}{n}\left(\sum o_i\right)^2}}$$

Template Matching

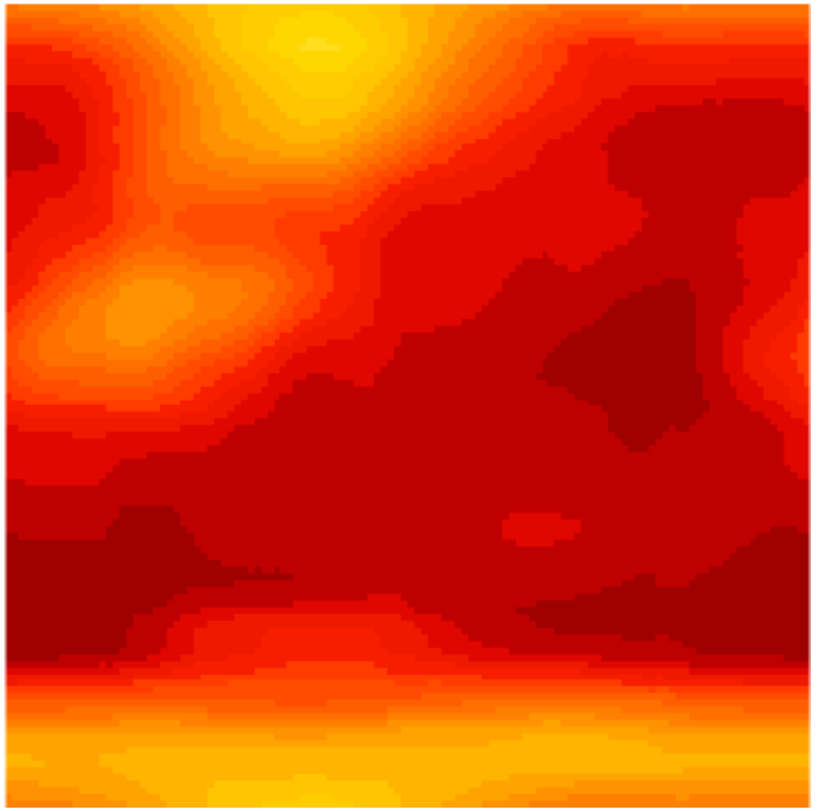
$\sum o_i = 40120$ $\sum o_i^2 = 4781080$

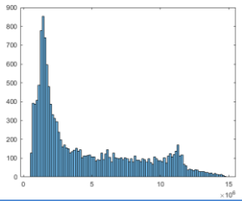
Here are our statistic images.



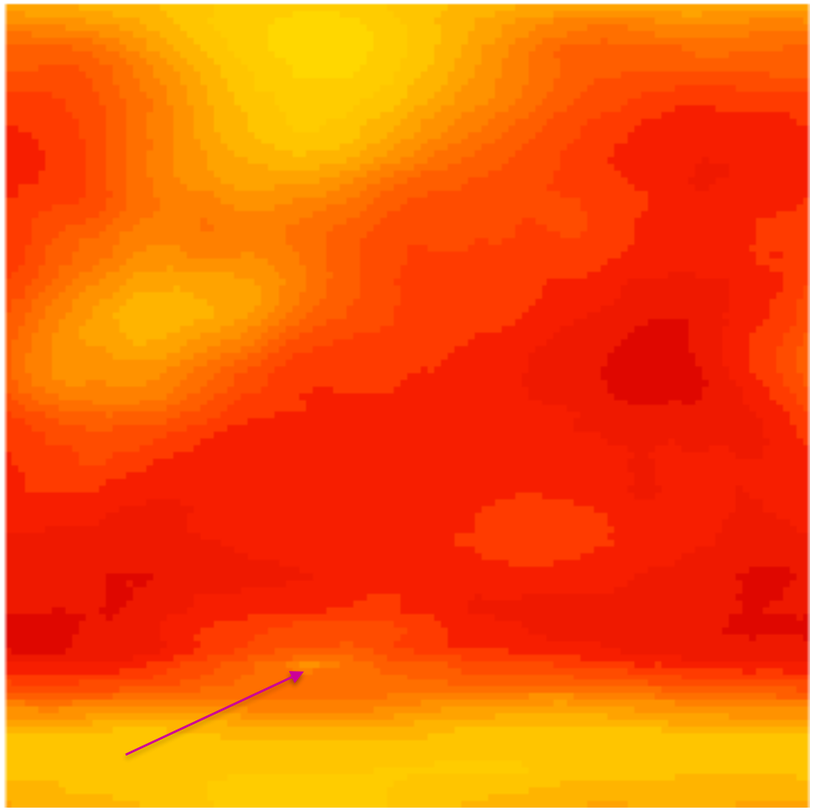


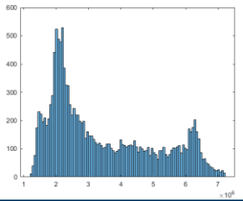
$\sum p_i$





$\sum p_i^2$

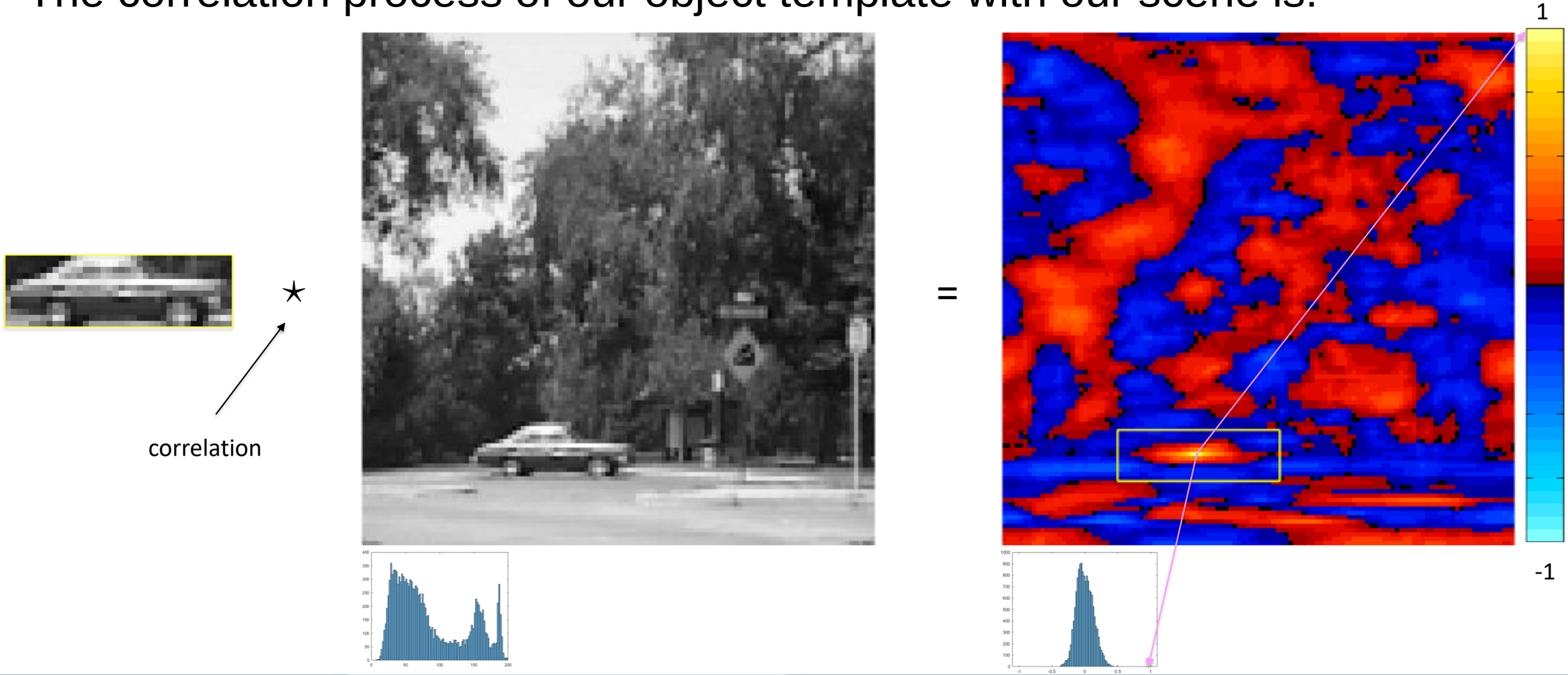




$\sum o_i p_i$

Template Matching

The correlation process of our object template with our scene is:



Through Image Processing

Dr. Daniel B. Rowe
Professor of Computational Statistics
Department of Mathematical and Statistical Sciences
Marquette University

Videos in Matlab

Original Images

General Properties:
Name: 'Cat.mp4'
Path: 'C:MATH4931
Duration: 24.8686
CurrentTime: 24.8686
NumFrames: 373

Video Properties:
Width: 1280
Height: 720
FrameRate: 15.0282
BitsPerPixel: 24
VideoFormat: 'RGB24'

Videos in Matlab

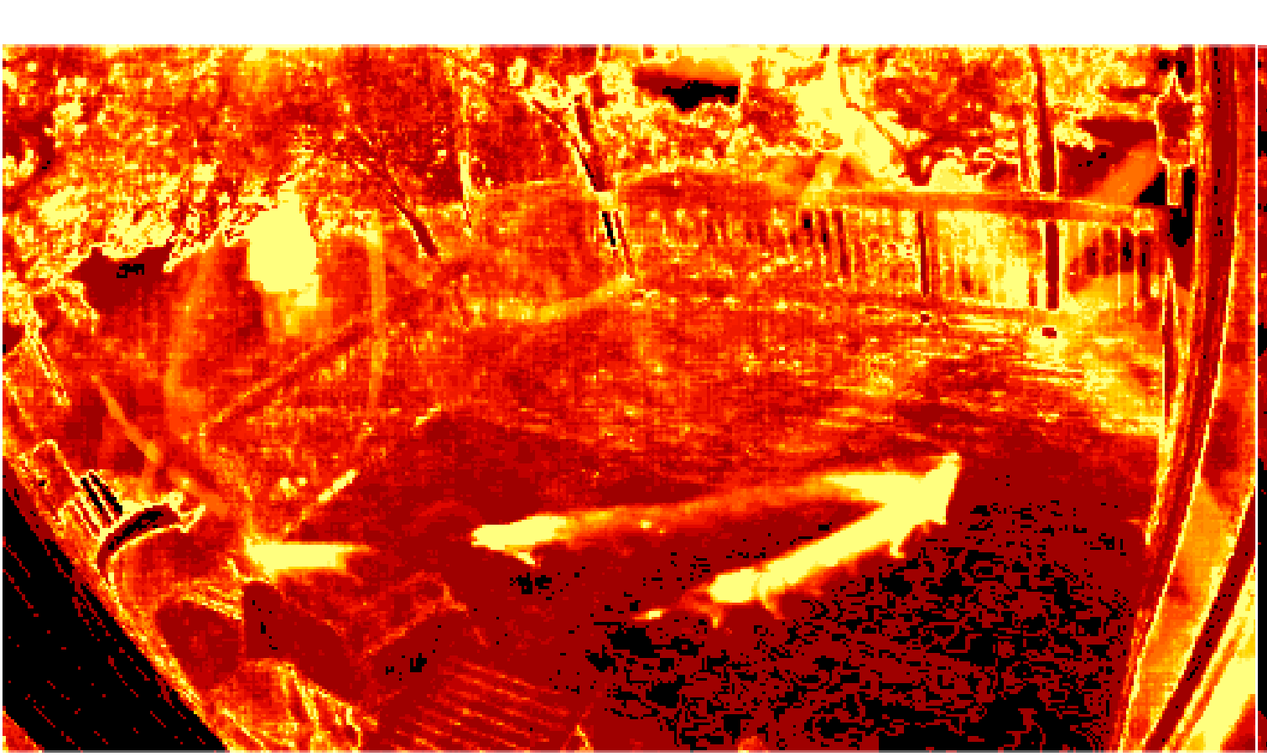


Image Mean

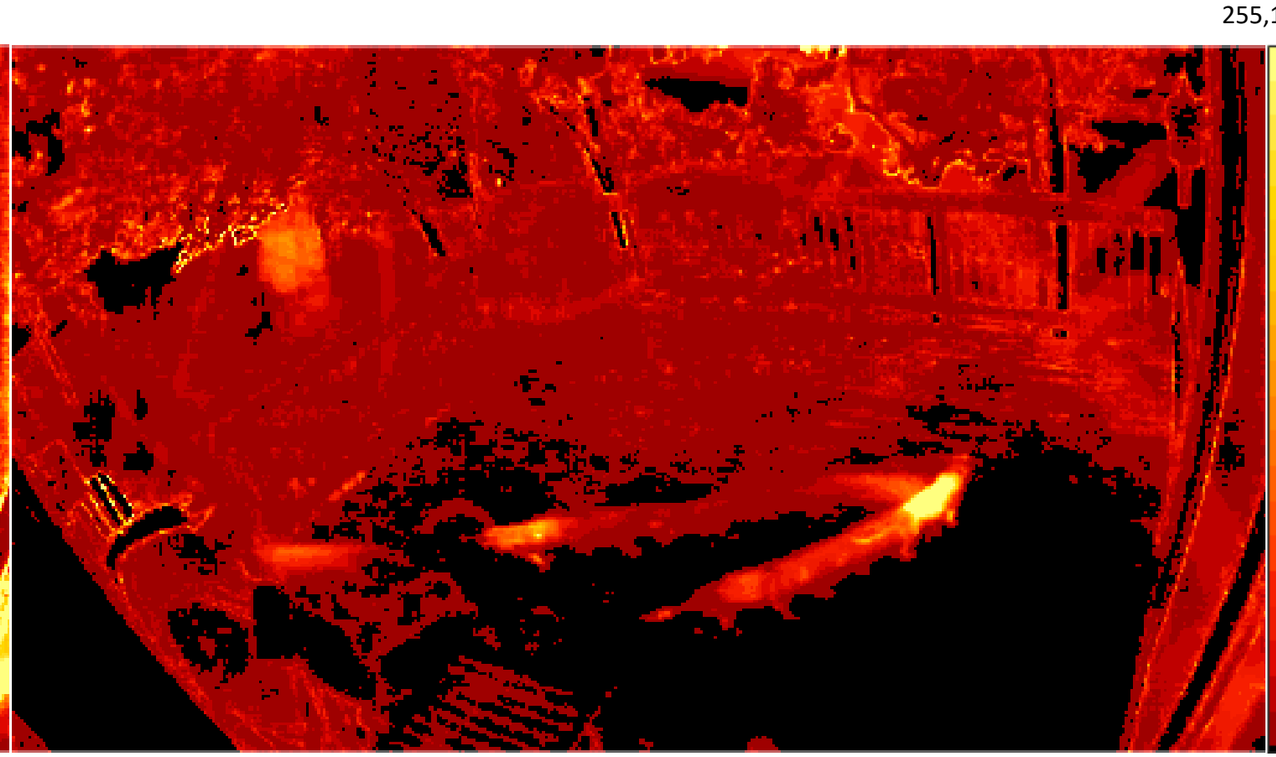
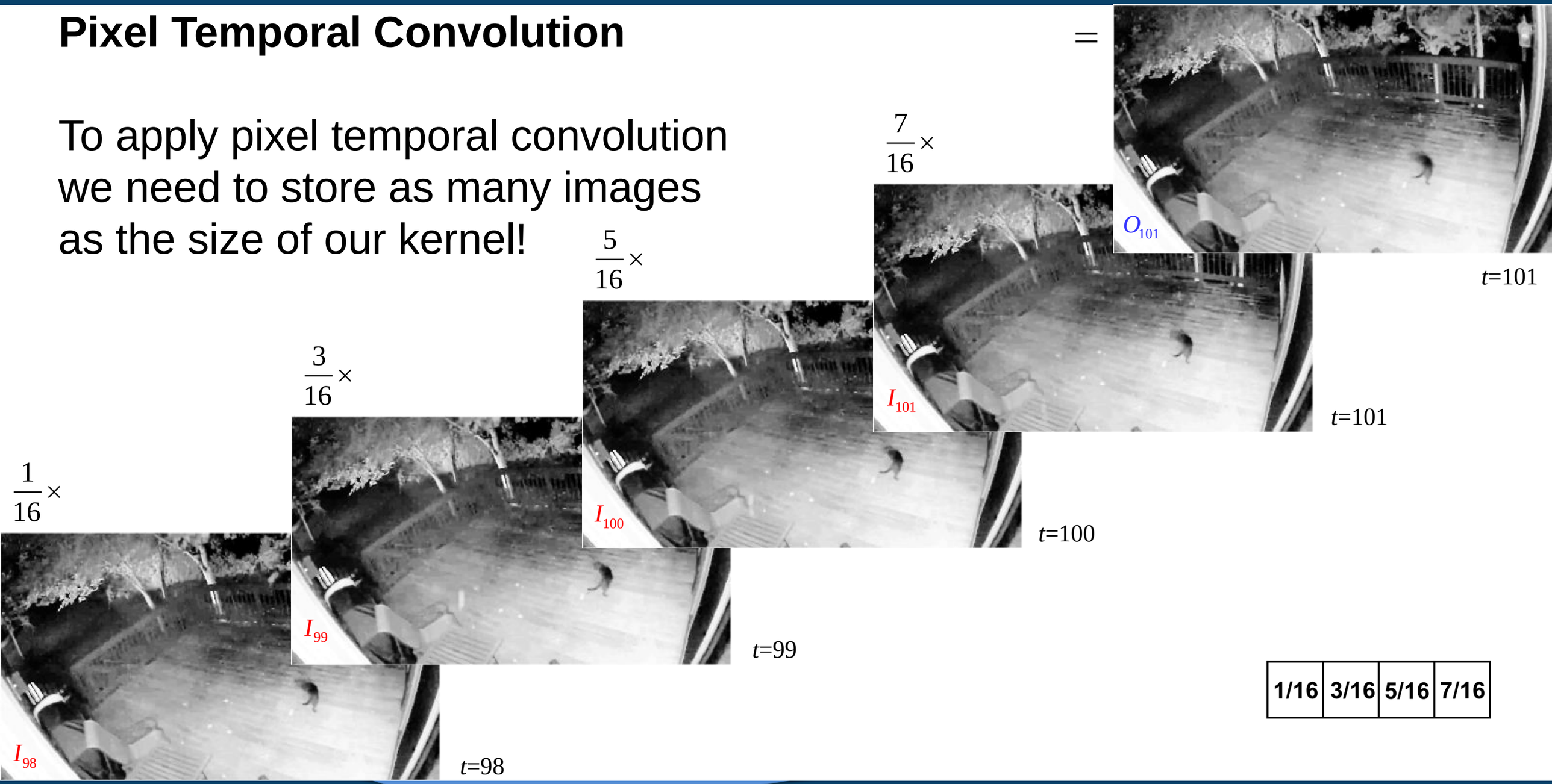


Image Variance

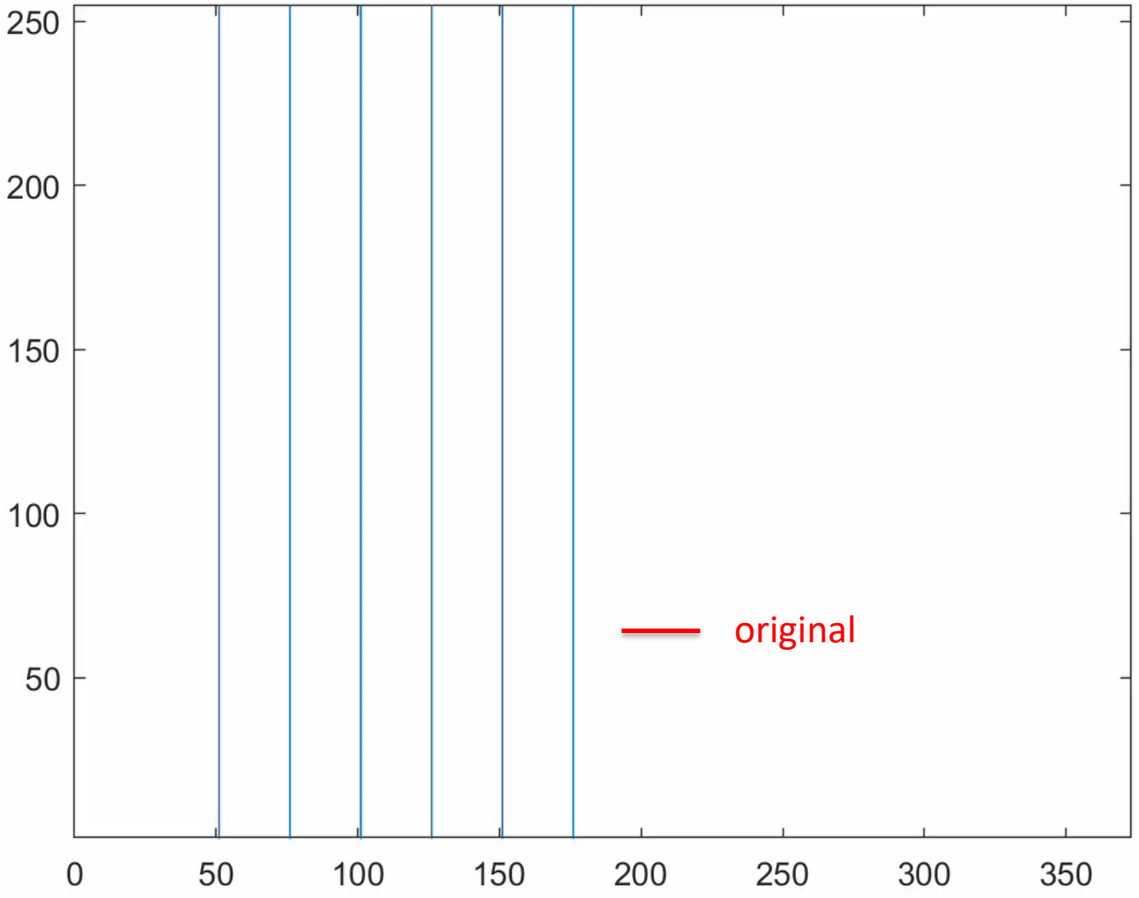
Pixel Temporal Convolution

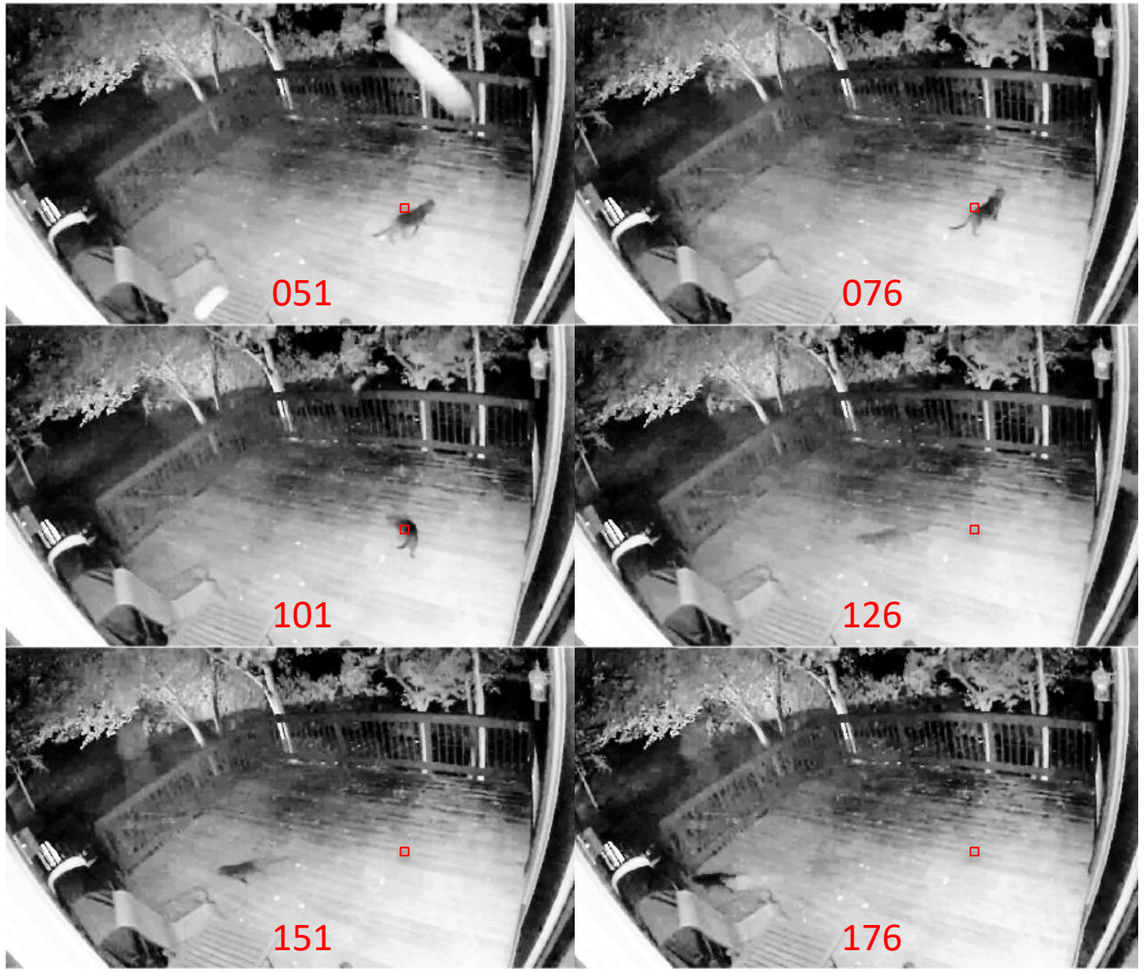
To apply pixel temporal convolution we need to store as many images as the size of our kernel!



Pixel Temporal Convolution

View the time series of a particular pixel.





The Discrete Fourier Transform

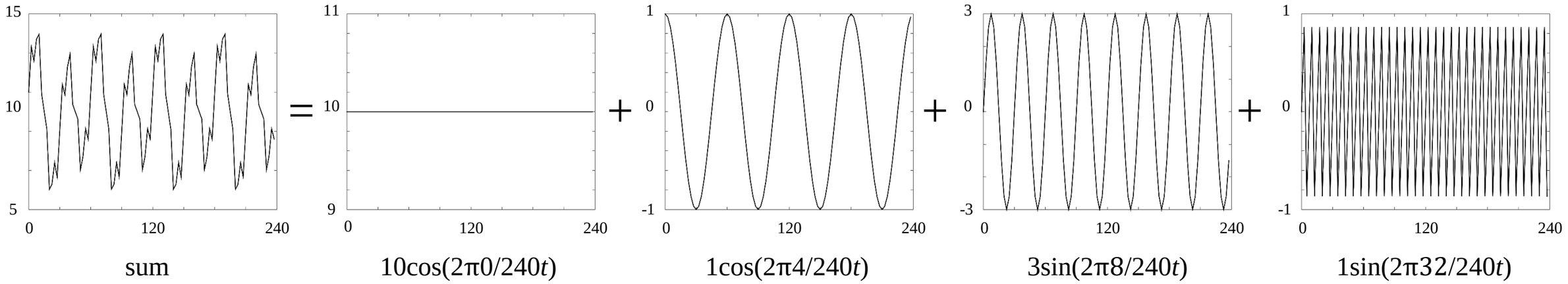
Dr. Daniel B. Rowe
Professor of Computational Statistics
Department of Mathematical and Statistical Sciences
Marquette University

1D Discrete Fourier Transform

Example: Let's sample the continuous time series (1D function)

$$y(t) = 10\cos\left(2\pi\frac{0}{240}t\right) + \cos\left(2\pi\frac{4}{240}t\right) + 3\sin\left(2\pi\frac{8}{240}t\right) + \sin\left(2\pi\frac{32}{240}t\right)$$

at $t=1\Delta t,2\Delta t,3\Delta t,\ldots,n\Delta t$, where $n=96$ and $\Delta t=2.5$s for a total time of 240s.



1D Discrete Fourier Transform

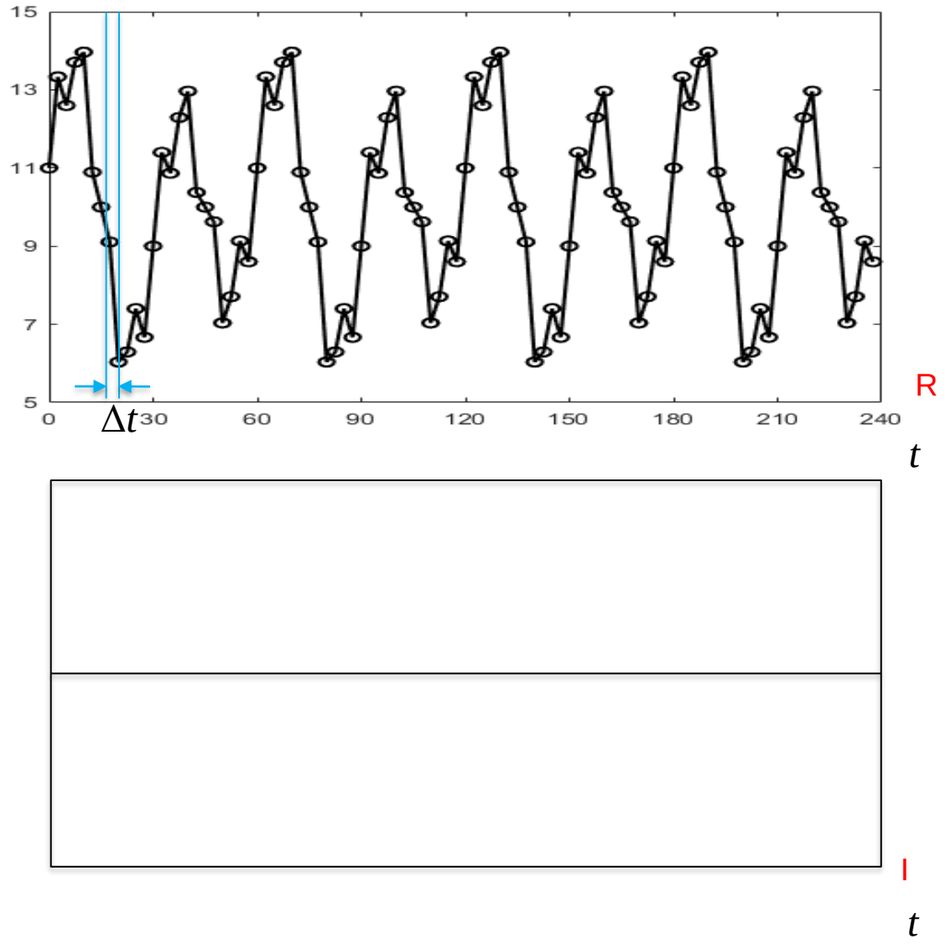
$n=96$

$\Delta t=2.5\text{s}$

$$\Delta\nu=\frac{1}{n\Delta t}=0.042Hz$$

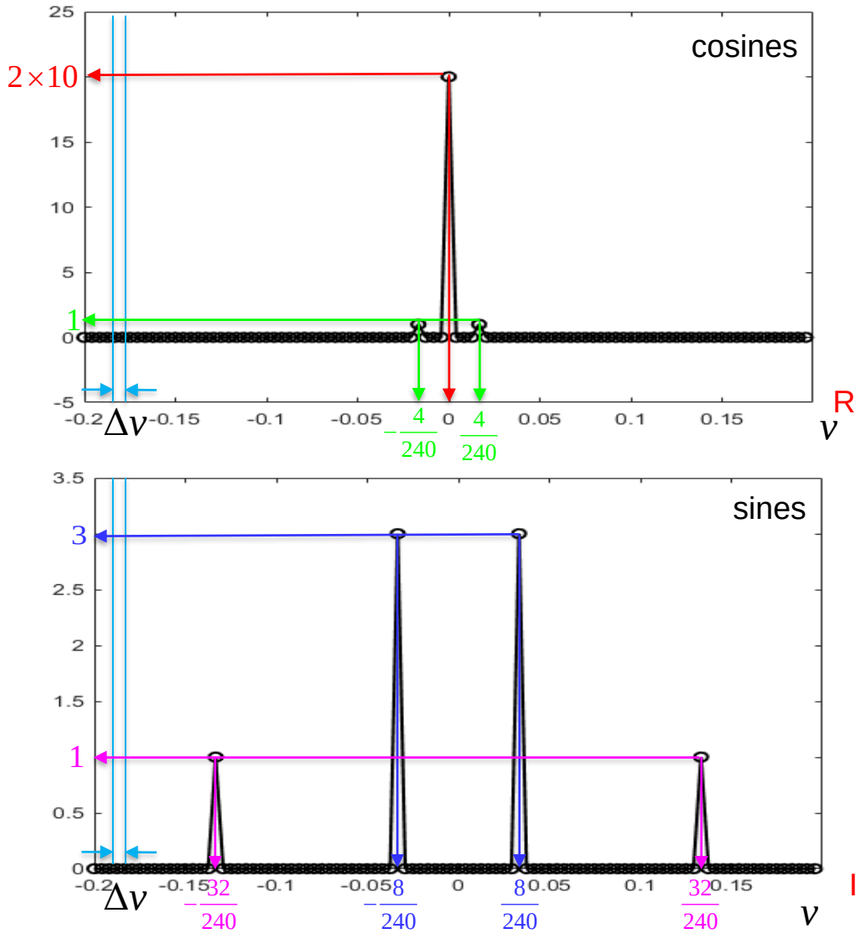
$$\nu_{\max}=\frac{n}{2}\Delta\nu=20Hz$$

Time Series



$\mathcal{F}$

Amplitude Coefficients

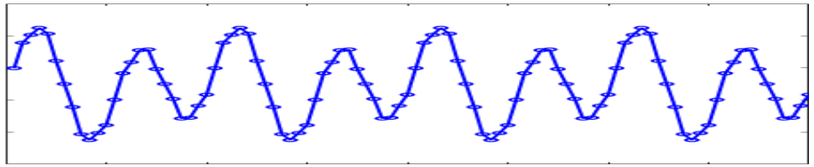


$$y(t)=10\cos\left(2\pi\frac{0}{240}t\right)+1\cos\left(2\pi\frac{4}{240}t\right)+3\sin\left(2\pi\frac{8}{240}t\right)+1\sin\left(2\pi\frac{32}{240}t\right)$$

*coefficients divided by $n/2$ for display

Time Series Convolution via the DFT

When we performed convolution of a time series with a kernel, we moved the kernel, multiplied, and created a new time point. This was repeated to create an entirely new time series.



smoothed

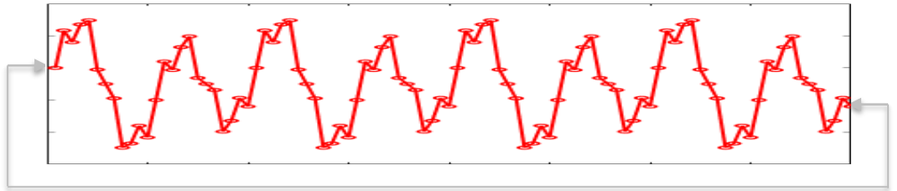
=

1/4	1/2	1/4

kernel

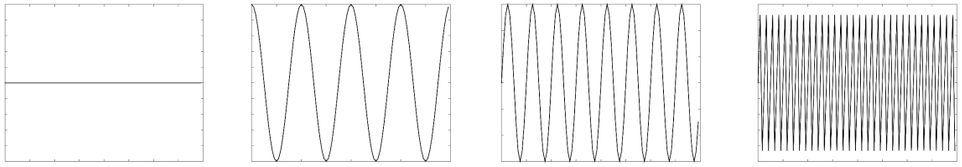
$*$

convolution



original

wrap

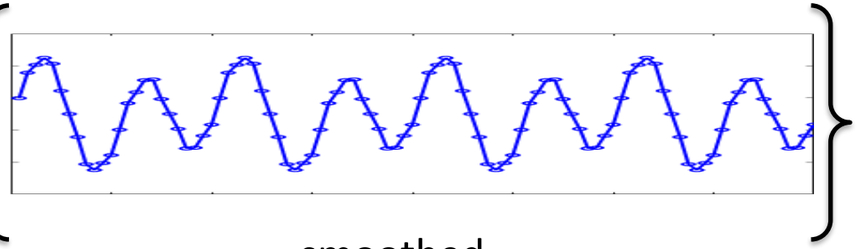


$$y(t) = 10\cos\left(2\pi\frac{0}{240}t\right) + 1\cos\left(2\pi\frac{4}{240}t\right) + 3\sin\left(2\pi\frac{8}{240}t\right) + 1\sin\left(2\pi\frac{32}{240}t\right)$$

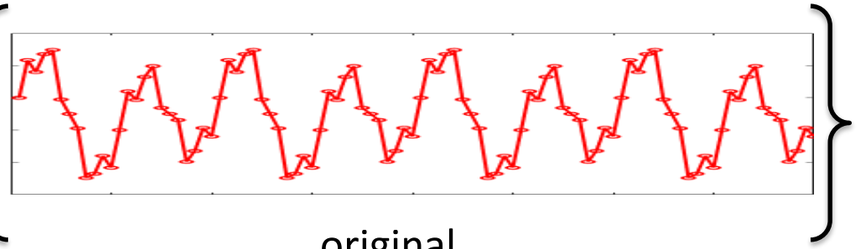
Time Series Convolution via the DFT

We can take the Fourier transform of the smoothed time series and compare it to the Fourier transform of the original input time series to see how the high frequency amplitude coefficients were attenuated.

1/4	1/2	1/4

$\mathcal{F}\{$  $\}$ =

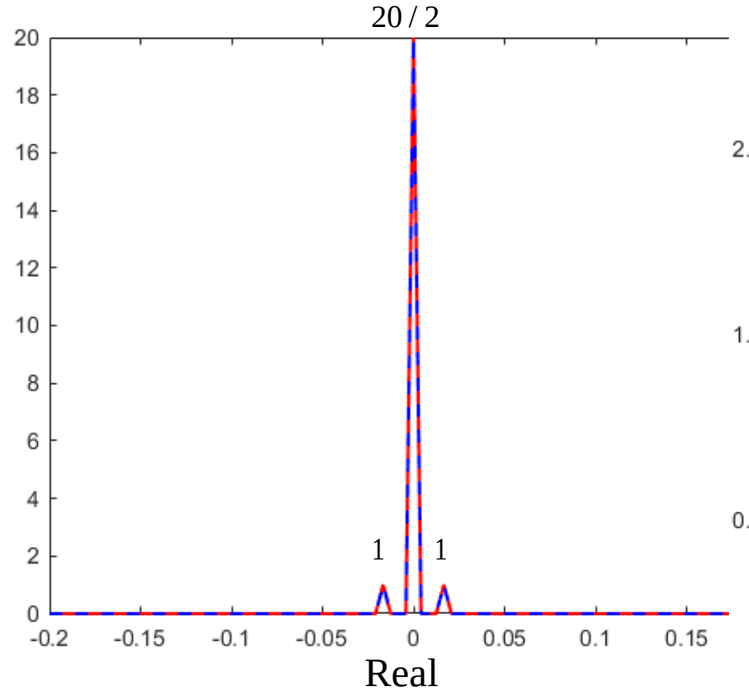
smoothed

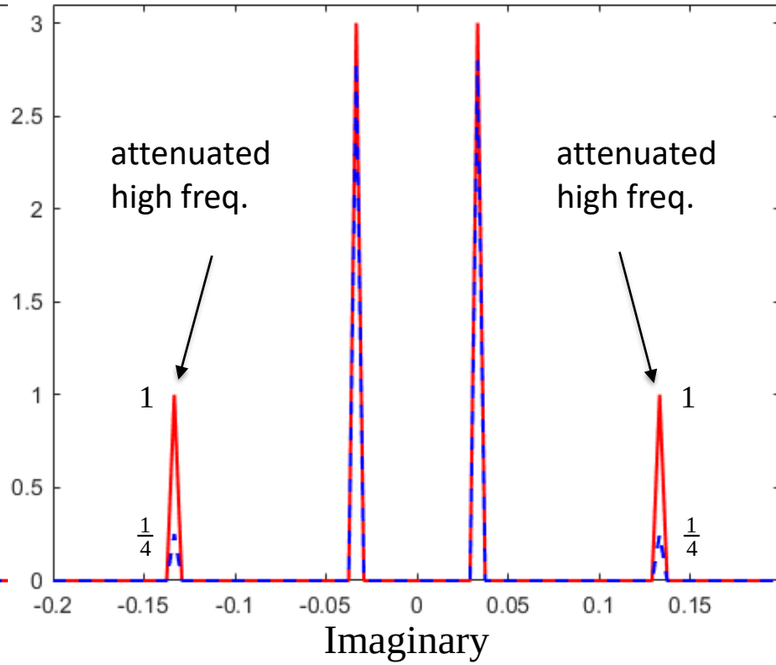
$\mathcal{F}\{$  $\}$ =

original

$$y(t) = 10\cos\left(2\pi\frac{0}{240}t\right) + 1\cos\left(2\pi\frac{4}{240}t\right) + 3\sin\left(2\pi\frac{8}{240}t\right) + 1\sin\left(2\pi\frac{32}{240}t\right)$$

$1\sin\left(2\pi\frac{32}{240}t\right)$



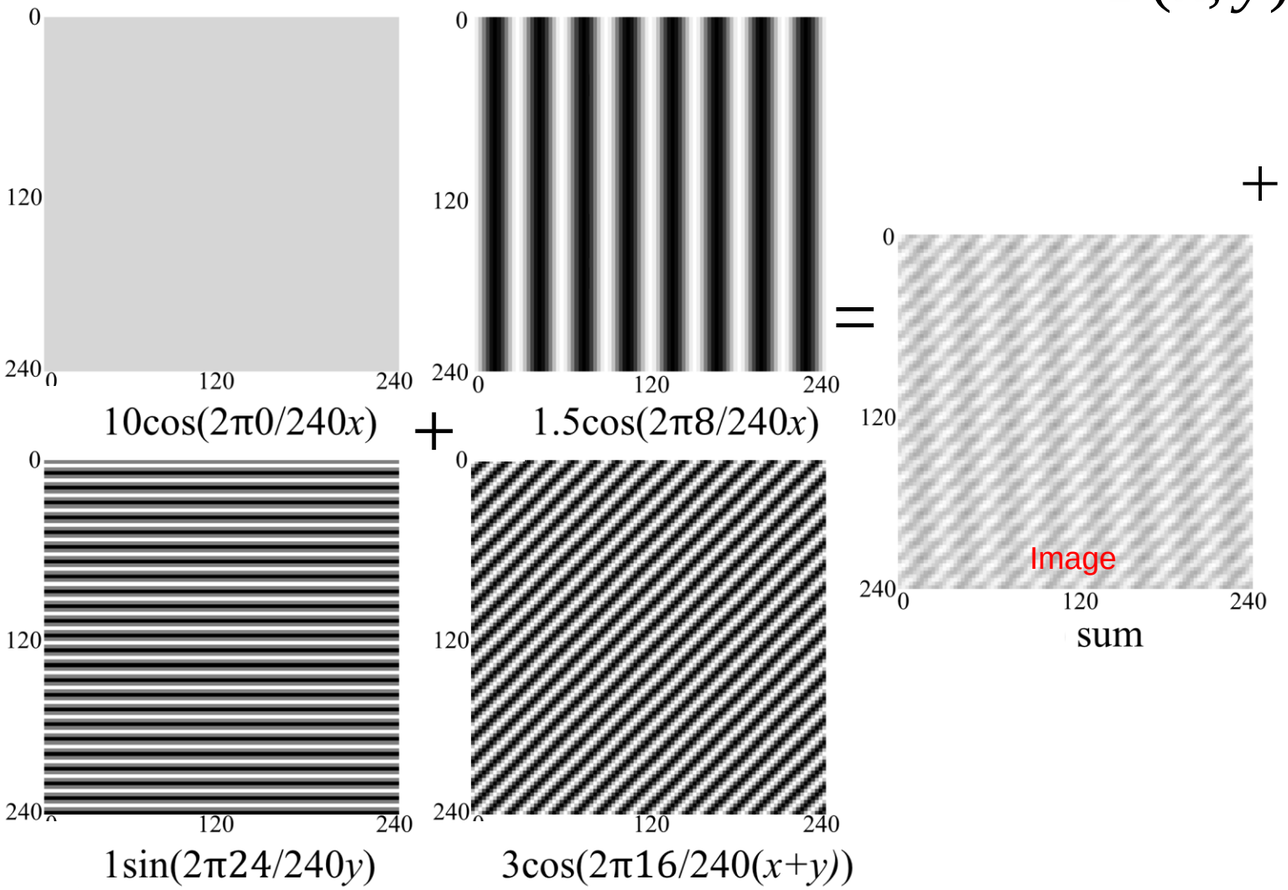


*coefficients divided by $n/2$ for display

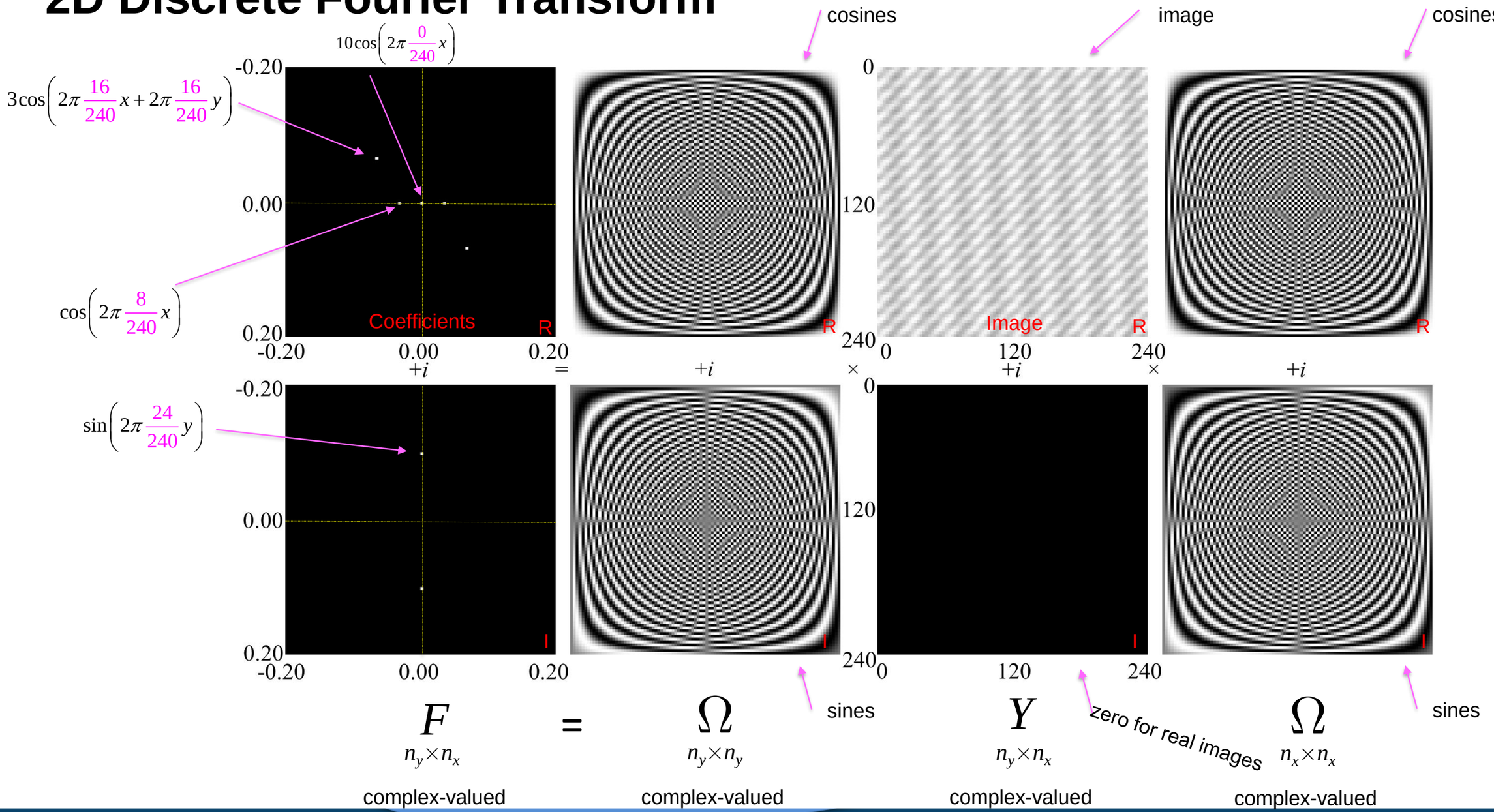
2D Discrete Fourier Transform

Example: Let's sample the continuous image scene (2D function)

$$Y(x,y) = 10\cos\left(2\pi\frac{0}{240}x\right) + \frac{3}{2}\cos\left(2\pi\frac{8}{240}x\right) + \sin\left(2\pi\frac{24}{240}y\right) + \cos\left(2\pi\left(\frac{16}{240}x + \frac{16}{240}y\right)\right)$$



2D Discrete Fourier Transform



Convolution via the DFT

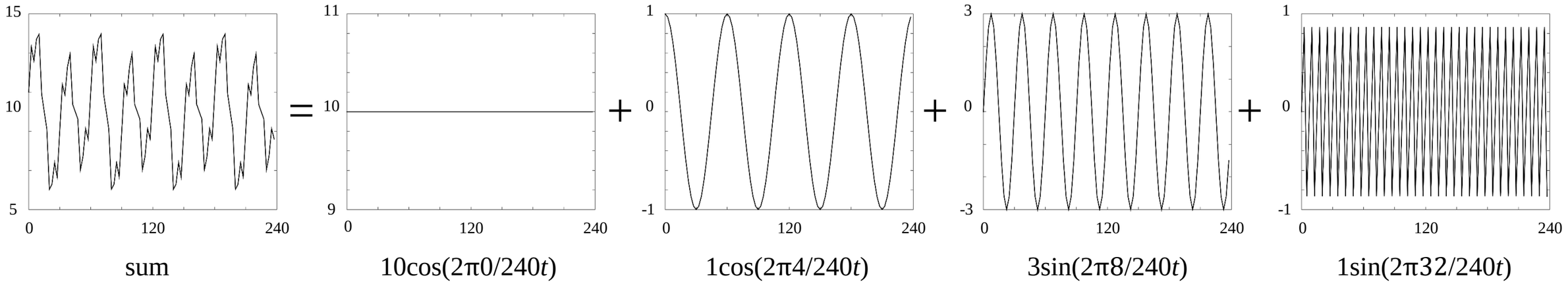
Dr. Daniel B. Rowe
Professor of Computational Statistics
Department of Mathematical and Statistical Sciences
Marquette University

Time Series Convolution via the DFT

Example: Let's sample the continuous time series (1D function)

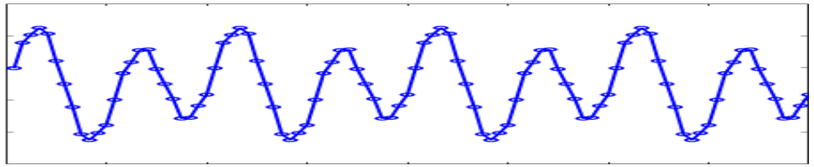
$$y(t) = 10\cos\left(2\pi\frac{0}{240}t\right) + \cos\left(2\pi\frac{4}{240}t\right) + 3\sin\left(2\pi\frac{8}{240}t\right) + \sin\left(2\pi\frac{32}{240}t\right)$$

at $t=1\Delta t,2\Delta t,3\Delta t,\ldots,n\Delta t$, where $n=96$ and $\Delta t=2.5$s for a total time of 240s.



Time Series Convolution via the DFT

When we performed convolution of a time series with a kernel, we moved the kernel, multiplied, and created a new time point. This was repeated to create an entirely new time series.



smoothed

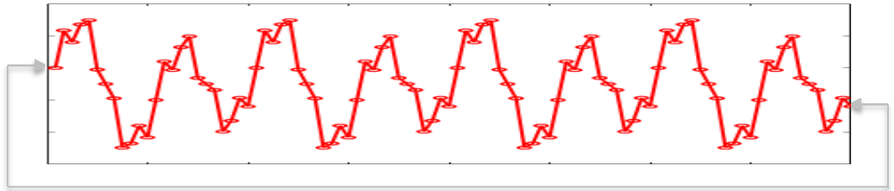
=

1/4	1/2	1/4

kernel

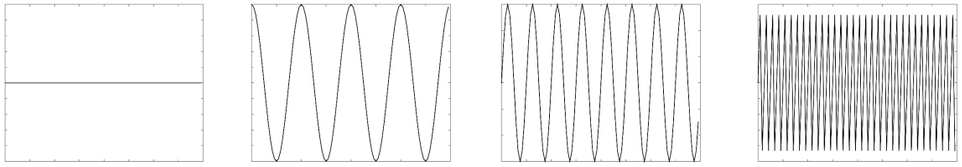
$*$

convolution



original

wrap

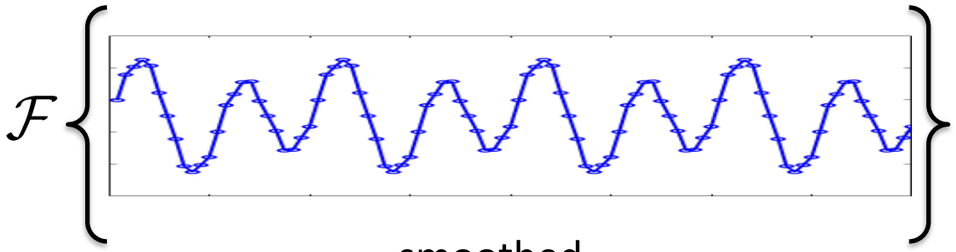


$$y(t) = 10\cos\left(2\pi\frac{0}{240}t\right) + 1\cos\left(2\pi\frac{4}{240}t\right) + 3\sin\left(2\pi\frac{8}{240}t\right) + 1\sin\left(2\pi\frac{32}{240}t\right)$$

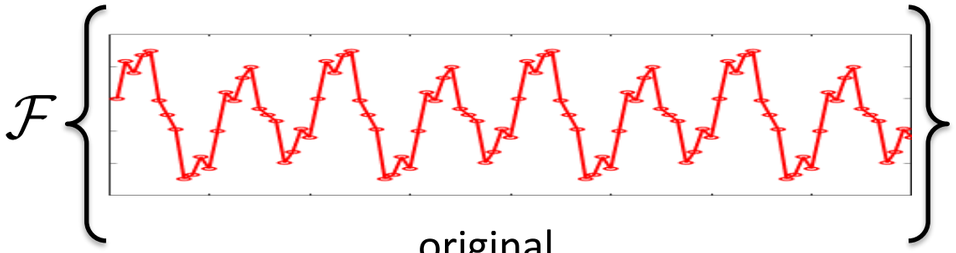
Time Series Convolution via the DFT

We can take the Fourier transform of the smoothed time series and compare it to the Fourier transform of the original input time series to see how the high frequency amplitude coefficients were attenuated.

1/4	1/2	1/4



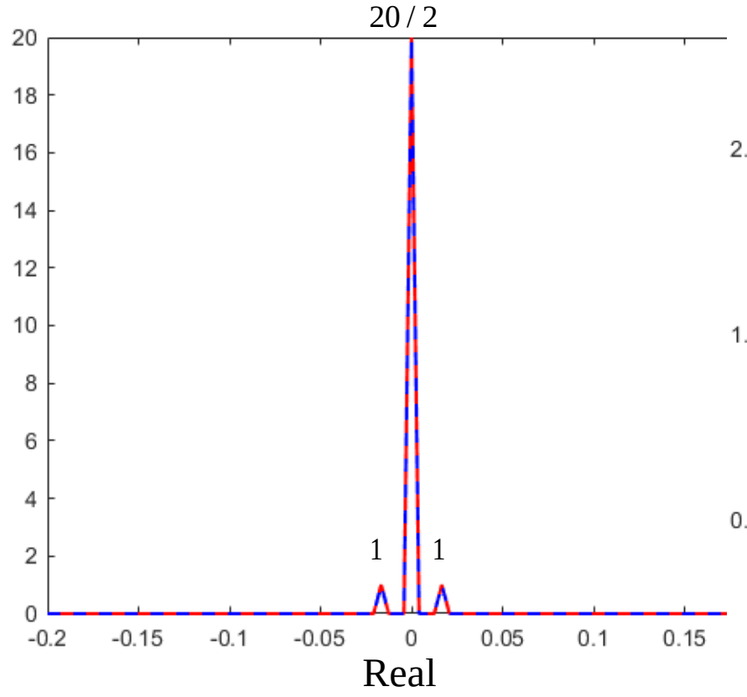
smoothed

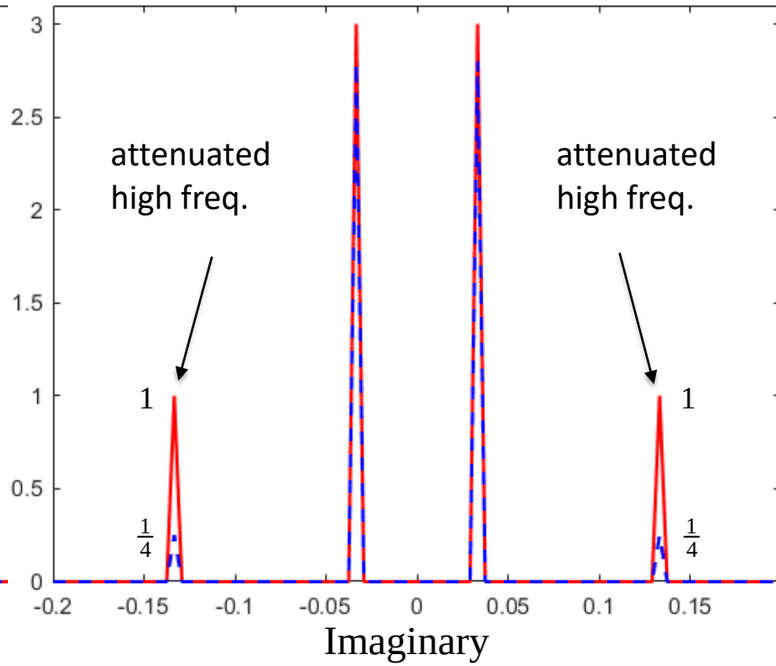


original

$$y(t) = 10\cos\left(2\pi\frac{0}{240}t\right) + 1\cos\left(2\pi\frac{4}{240}t\right) + 3\sin\left(2\pi\frac{8}{240}t\right) + 1\sin\left(2\pi\frac{32}{240}t\right)$$

$$1\sin\left(2\pi\frac{32}{240}t\right)$$



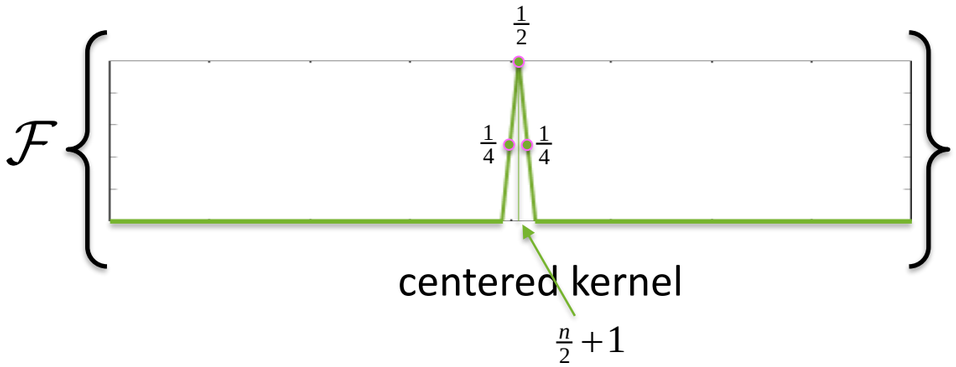


*coefficients divided by $n/2$ for display

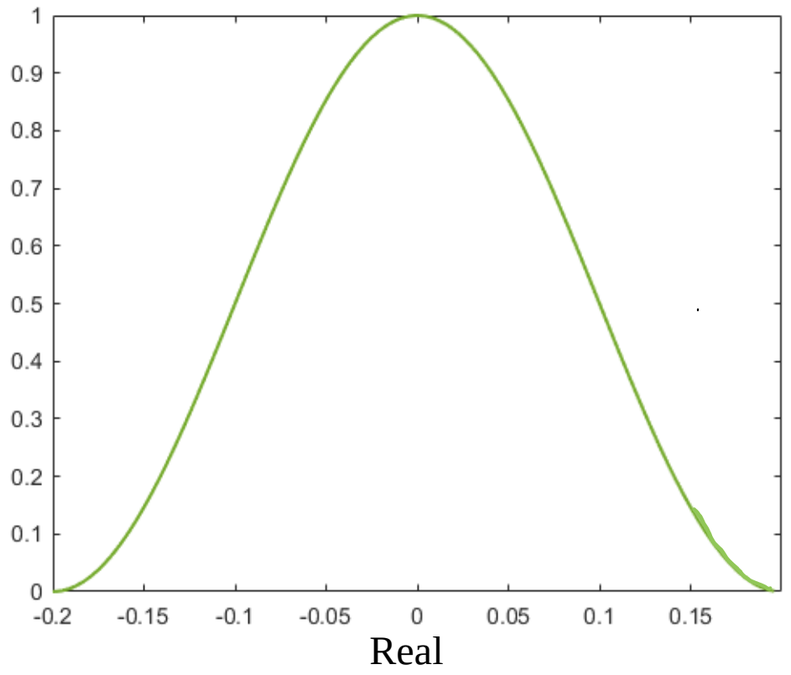
Time Series Convolution via the DFT

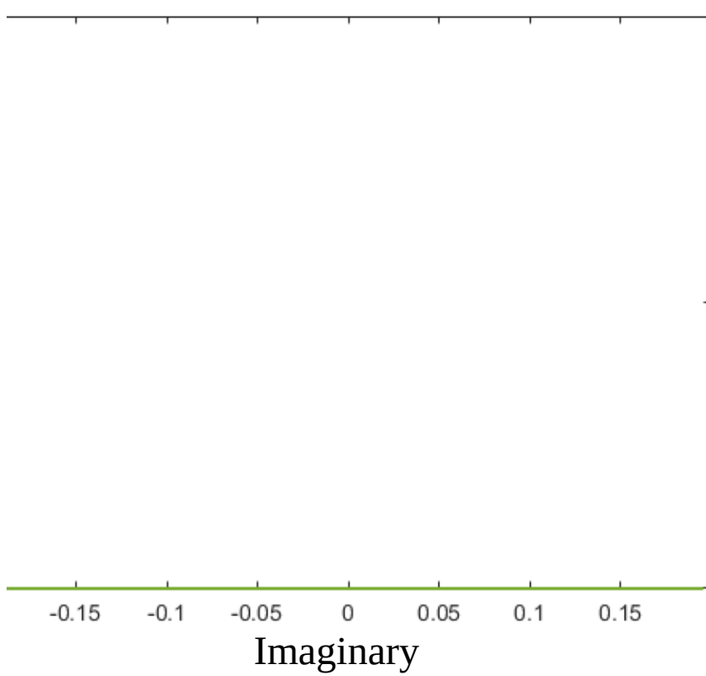
To demonstrate the convolution theorem, we take the discrete Forward Fourier transform of the centered kernel …

1/4	1/2	1/4



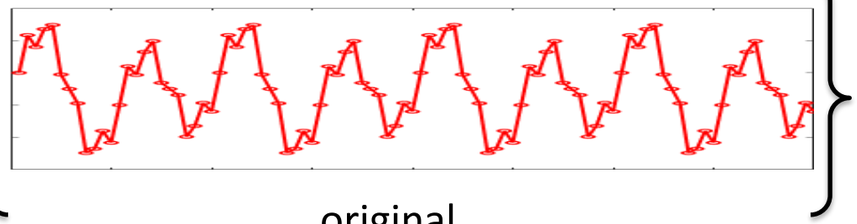
=



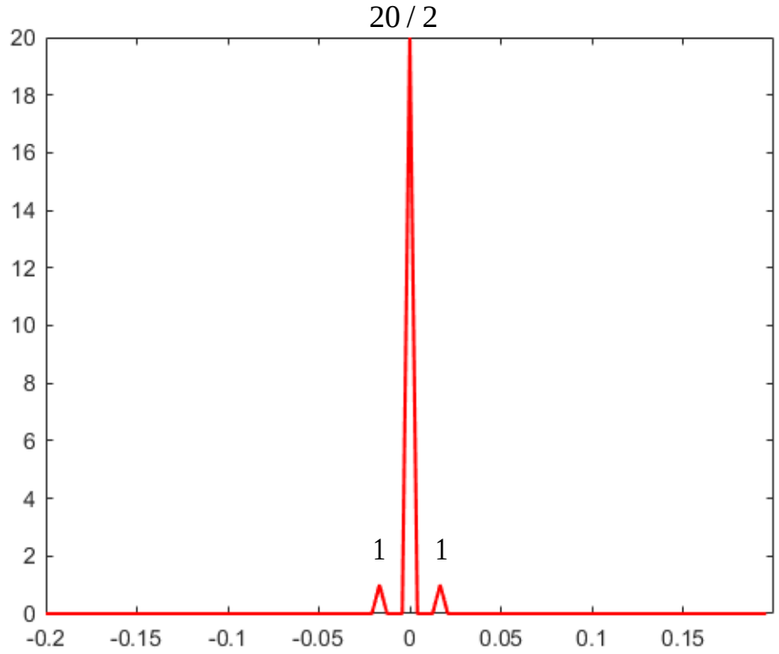


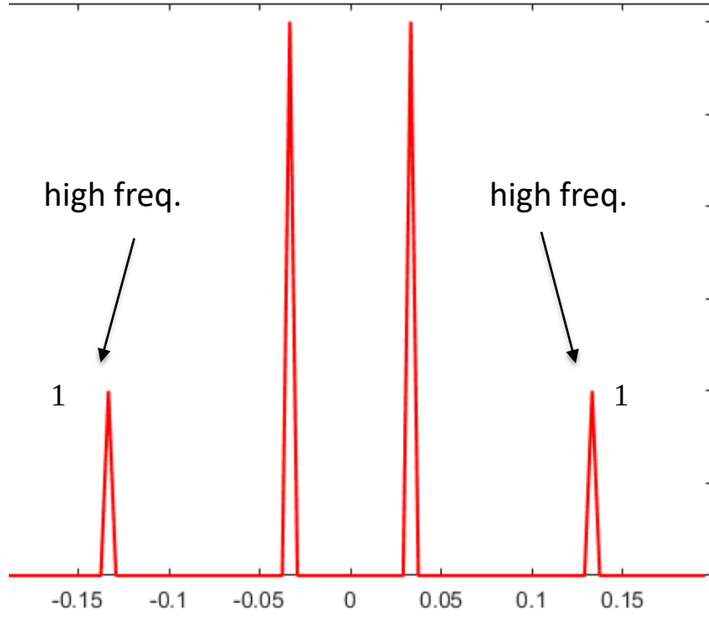
Time Series Convolution via the DFT

… and the forward discrete Fourier transform of the original input time series …

$\mathcal{F}\{$  $\}$ =

original





$$y(t) = 10\cos\left(2\pi\frac{0}{240}t\right) + 1\cos\left(2\pi\frac{4}{240}t\right) + 3\sin\left(2\pi\frac{8}{240}t\right) + 1\sin\left(2\pi\frac{32}{240}t\right)$$

*coefficients divided by $n/2$ for display

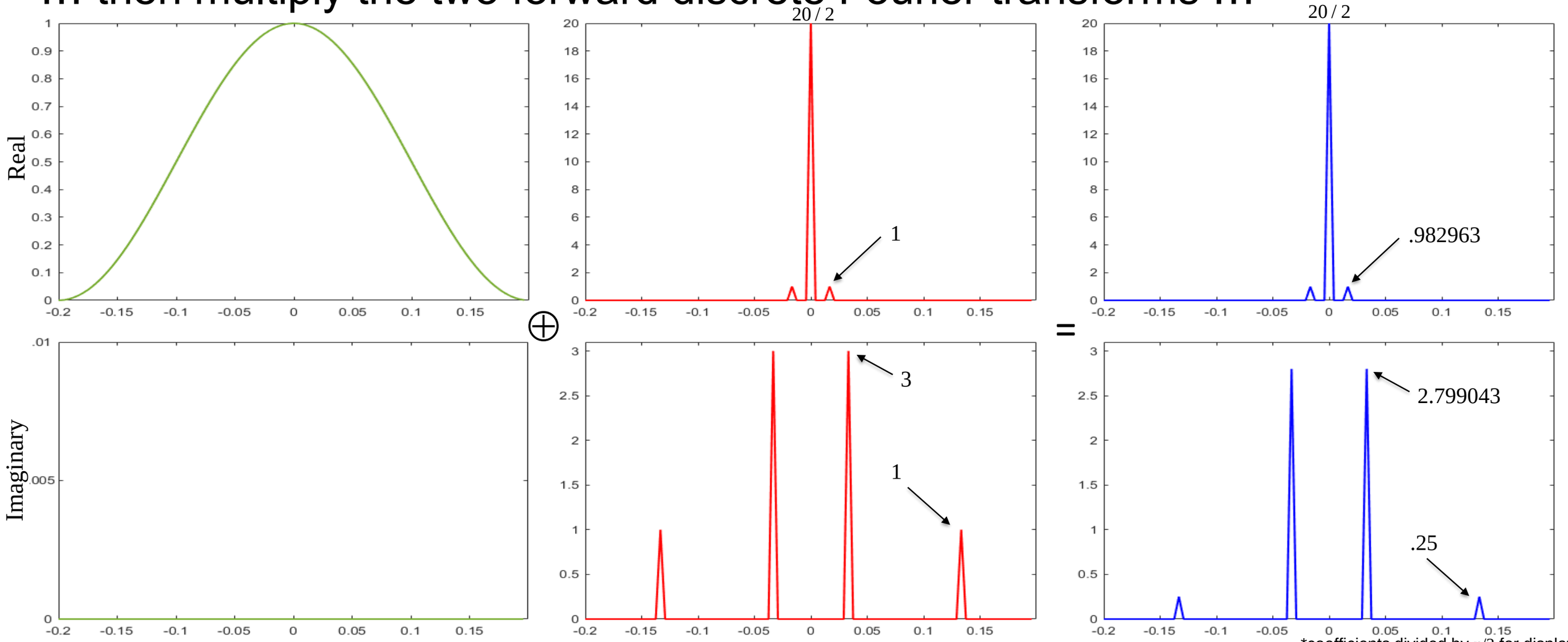
Time Series Convolution via the DFT

$$y(t) = 10\cos\left(2\pi\frac{0}{240}t\right) + 1\cos\left(2\pi\frac{4}{240}t\right) + 3\sin\left(2\pi\frac{8}{240}t\right) + 1\sin\left(2\pi\frac{32}{240}t\right)$$

⊕ = direct product, element -wise complex multiplication

".*" in Matlab

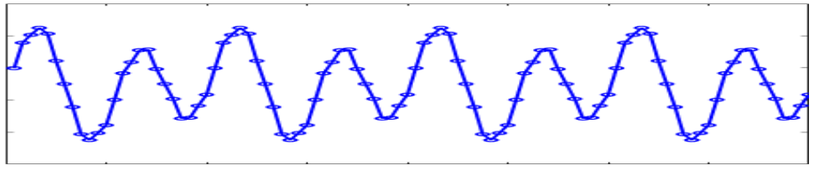
… then multiply the two forward discrete Fourier transforms …



*coefficients divided by $n/2$ for display

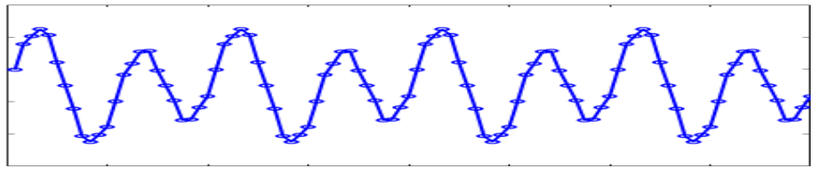
Time Series Convolution via the DFT

… and inverse discrete Fourier transform to obtain the smoothed time series.



FT smoothed

identical

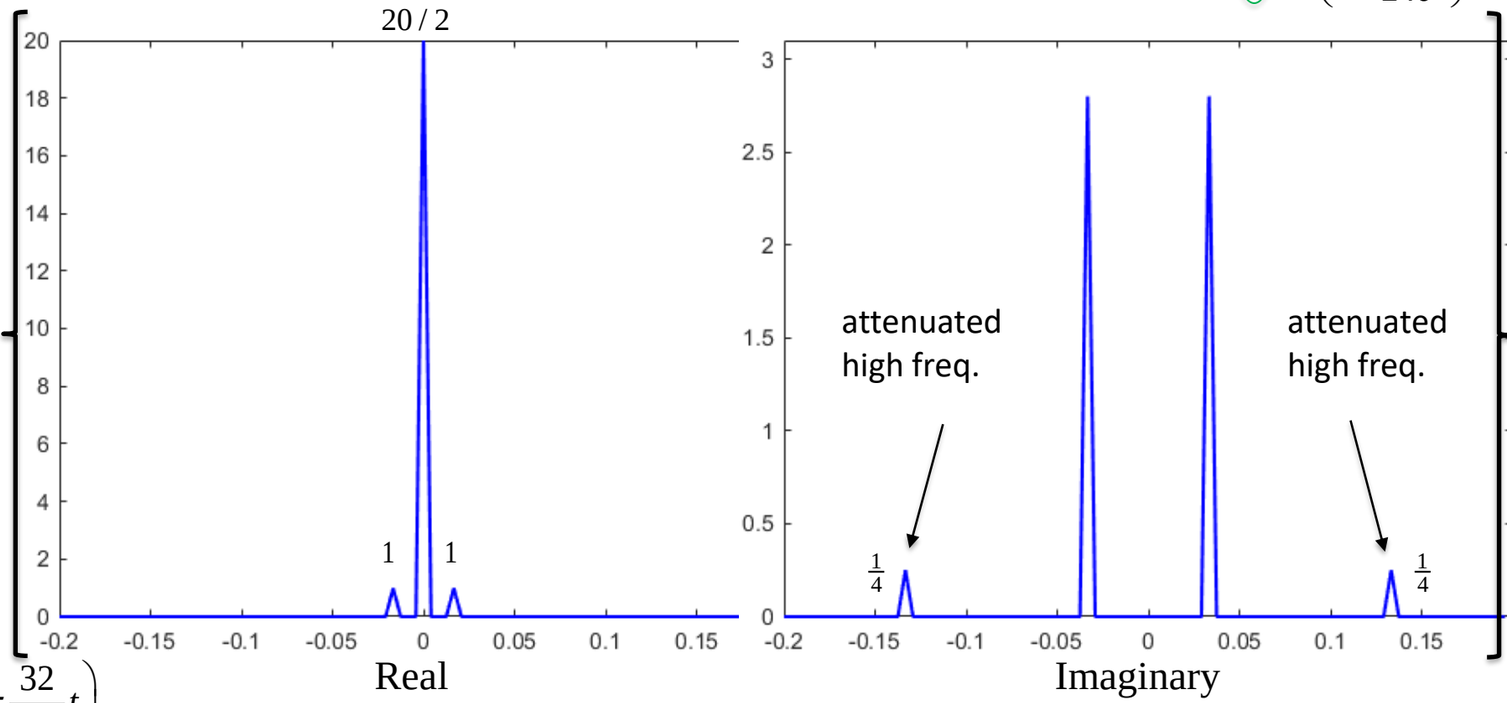


Convolution smoothed

=

$\mathcal{F}^{-1}$

$$\sin\left(2\pi\frac{32}{240}t\right)$$

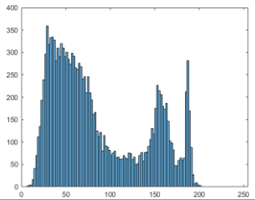


$$y(t)=10\cos\left(2\pi\frac{0}{240}t\right)+.98\cos\left(2\pi\frac{4}{240}t\right)+2.8\sin\left(2\pi\frac{8}{240}t\right)+.25\sin\left(2\pi\frac{32}{240}t\right)$$

*coefficients divided by $n/2$ for display

Image Convolution Example

We can use convolution in image space to compute a local weighted mean.



$\overline{x}_5$

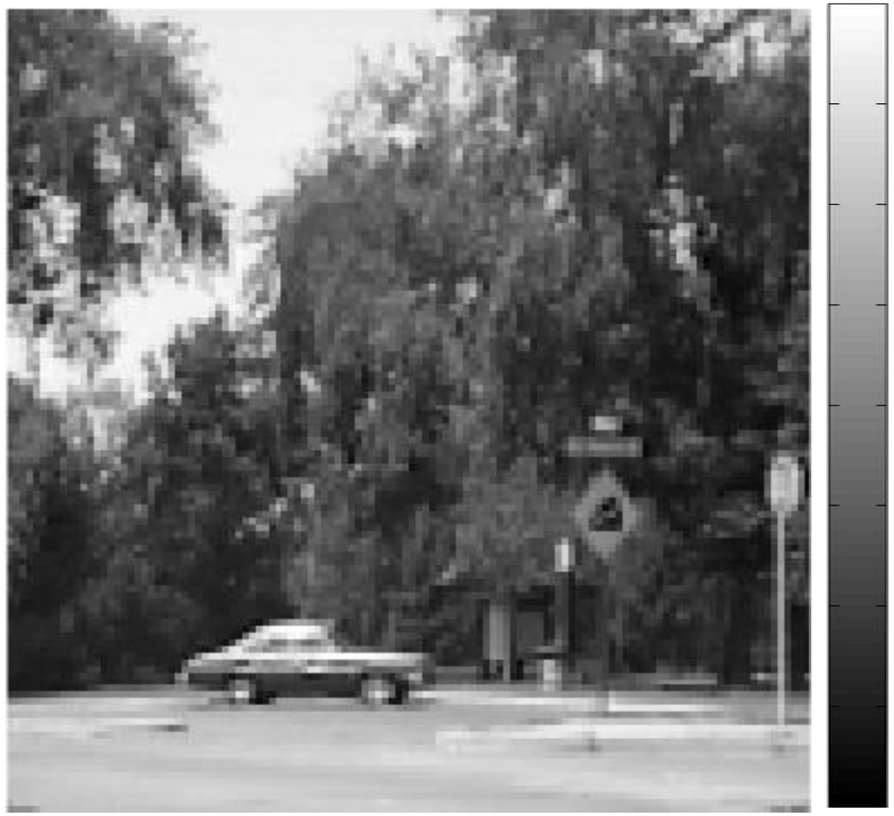
=

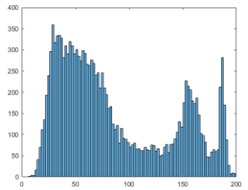
0	1/5	0
1/5	1/5	1/5
0	1/5	0

Kernel

$*$

convolution



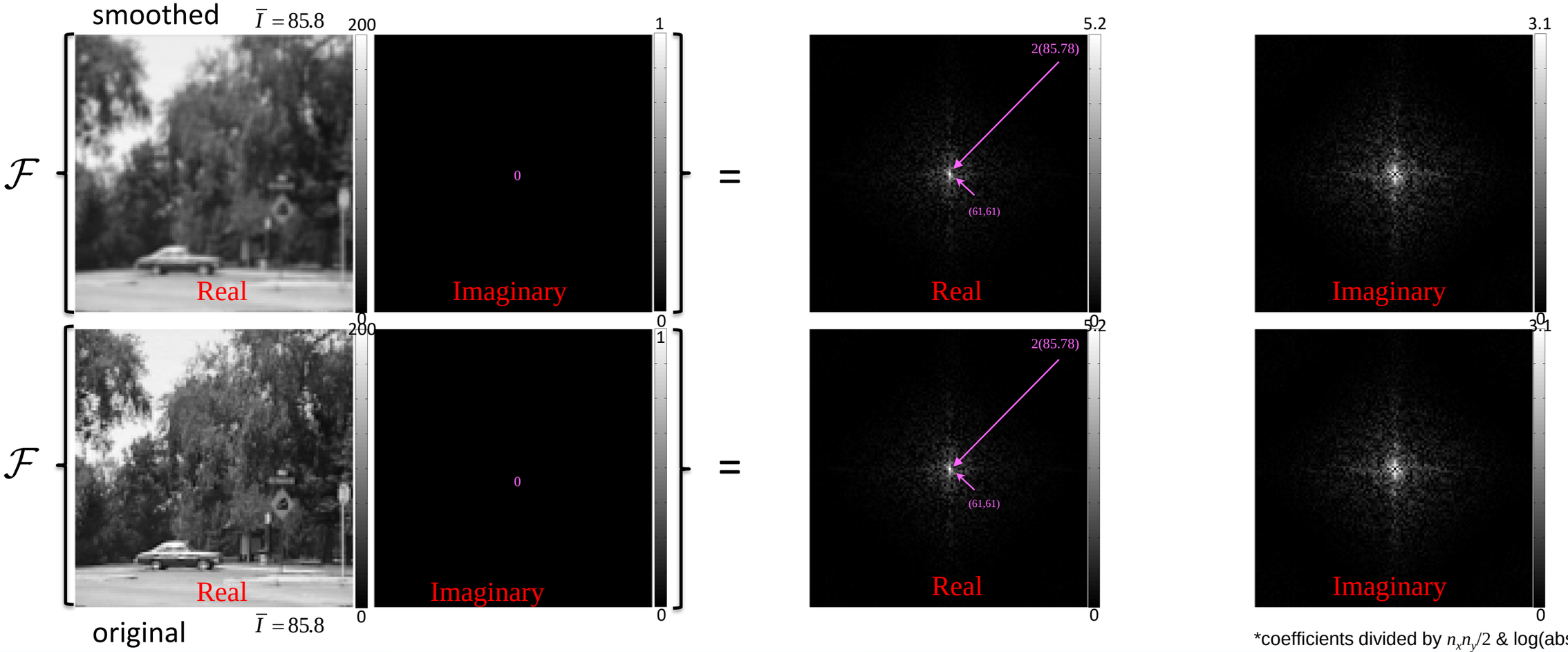


x_i

Image Convolution Example

120×120 image

We can Fourier transform of the smoothed image and compare it to that of the original input image to see high frequency coefficient attenuation.



*coefficients divided by $n_x n_y/2$ & log(abs(f)+1) for display

Image Convolution Example

⊕ = direct product, element -wise complex multiplication

… then multiply the two forward discrete Fourier transform together …

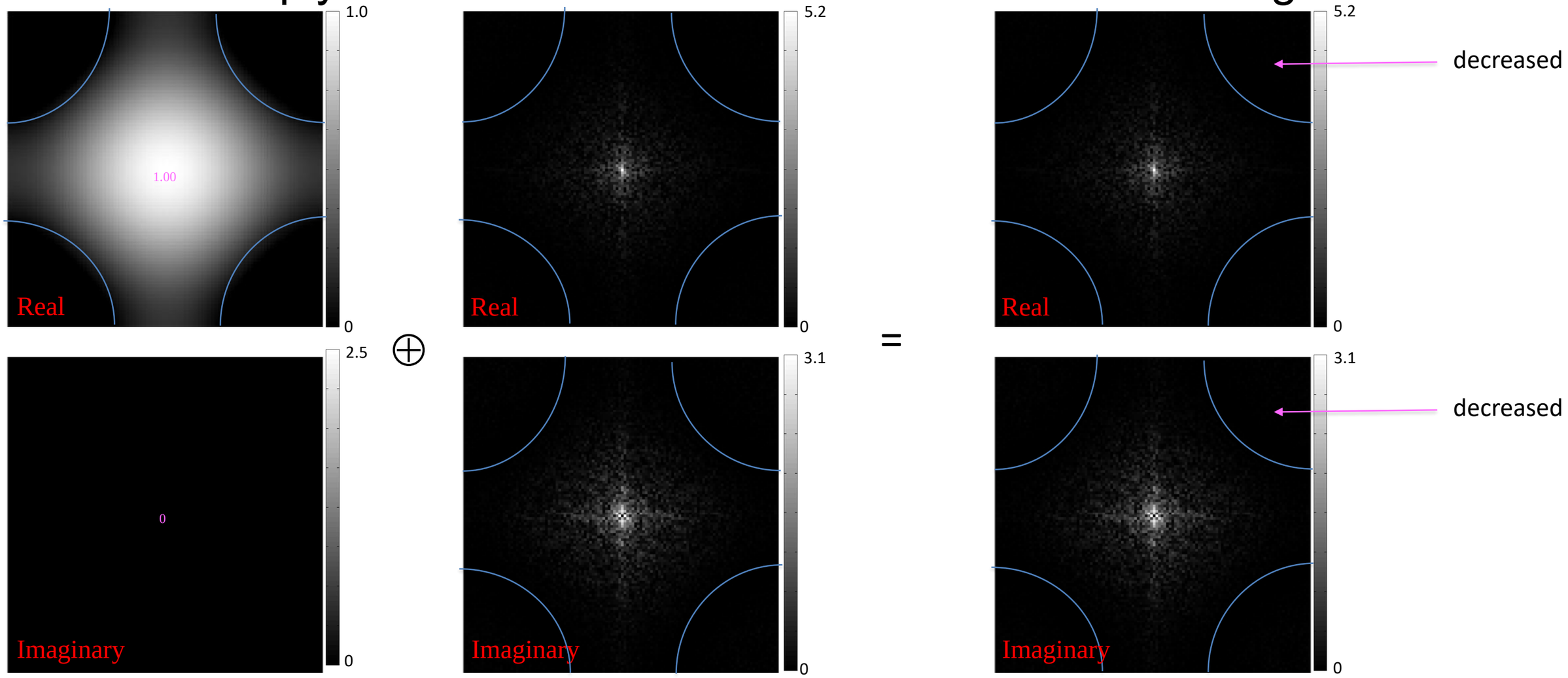
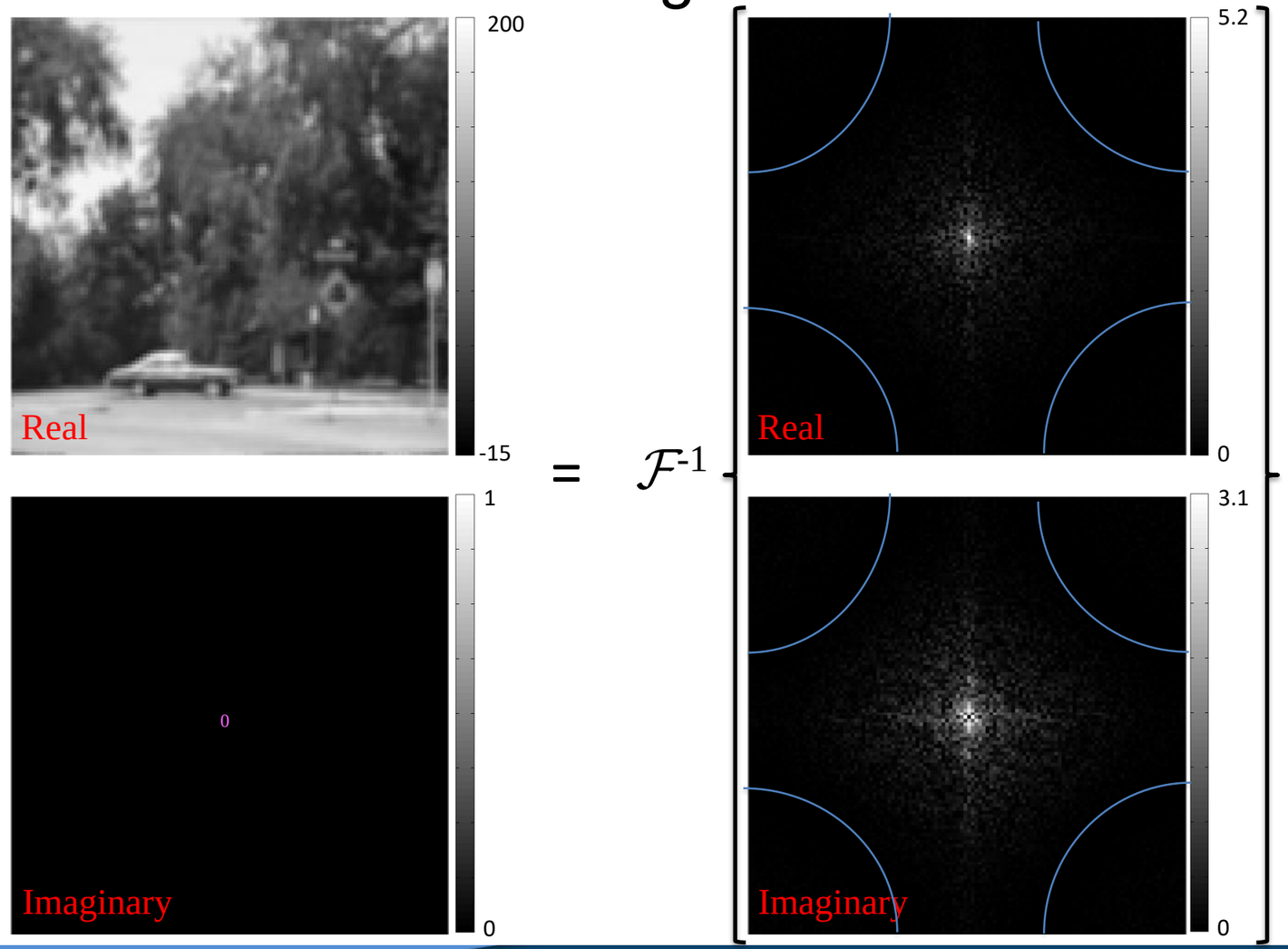


Image Convolution Example

… and inverse discrete Fourier transform to get our smoothed image.

This process yields the same result as convolution of the image with a kernel in image assuming wrap around!



Fast Object Tracking

Dr. Daniel B. Rowe
Professor of Computational Statistics
Department of Mathematical and Statistical Sciences
Marquette University

Template Matching via DFT

$$s^2 = \frac{\sum x_i^2 - \left(\sum x_i\right)^2 / n}{n-1}$$

We are going to use the DFT property to track cars with template matching.

For our object template we can pre-compute

$\sum o_i$ $\sum o_i^2$

pre-compute once

compute each frame for all neighborhoods

Then using the DFT compute

$\sum p_i$ $\sum p_i^2$ $\sum o_i p_i$

$$r = \frac{\sum p_i o_i - \frac{1}{n}\left(\sum p_i\right)\left(\sum o_i\right)}{\sqrt{\sum p_i^2 - \frac{1}{n}\left(\sum p_i\right)^2}\sqrt{\sum o_i^2 - \frac{1}{n}\left(\sum o_i\right)^2}}$$

Template Matching via DFT

We now have all the pieces that we need.

$\sum o_i = 10358226$

$\sum o_i^2 = 1929989924$

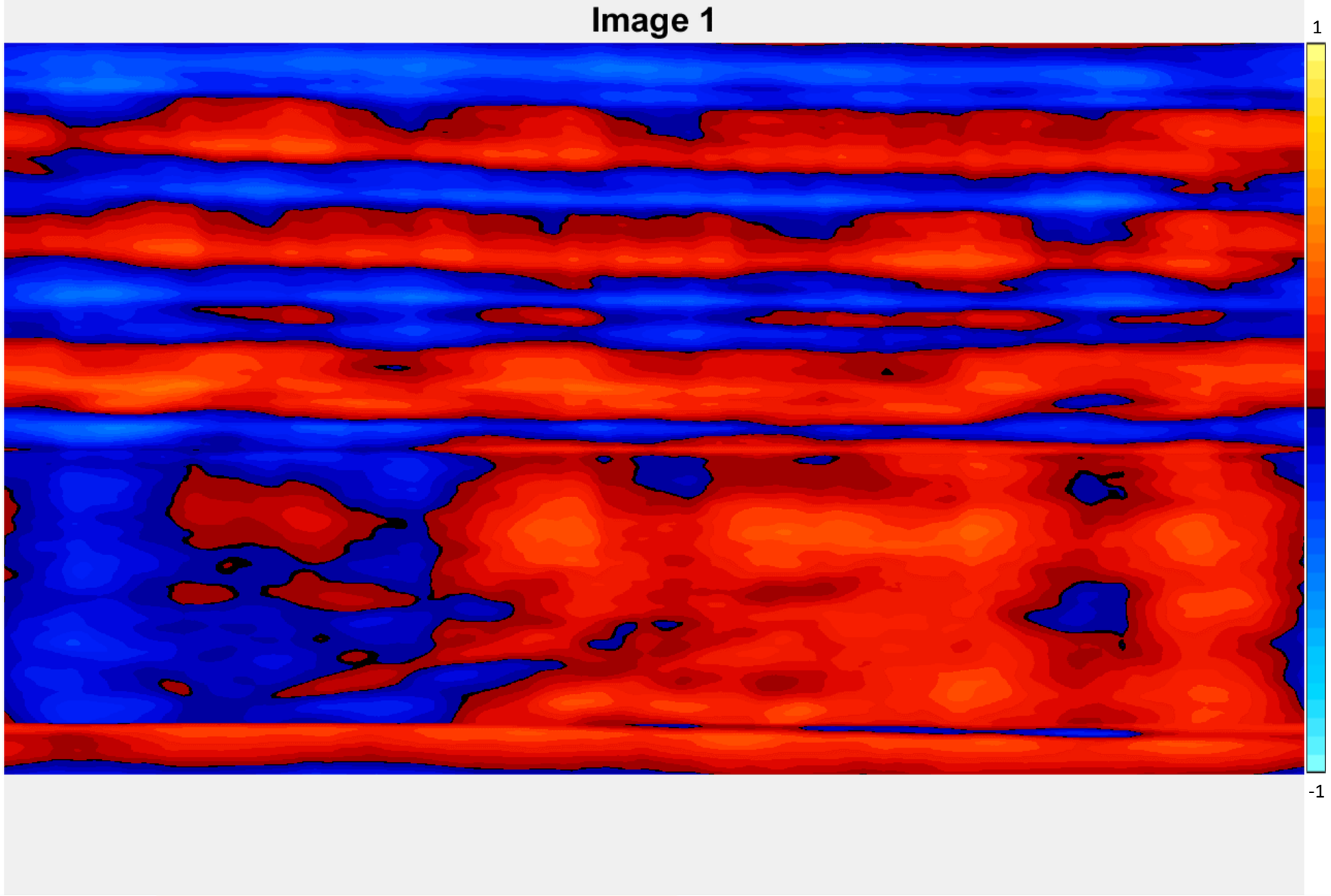
$\sum p_i$

$\sum p_i^2$

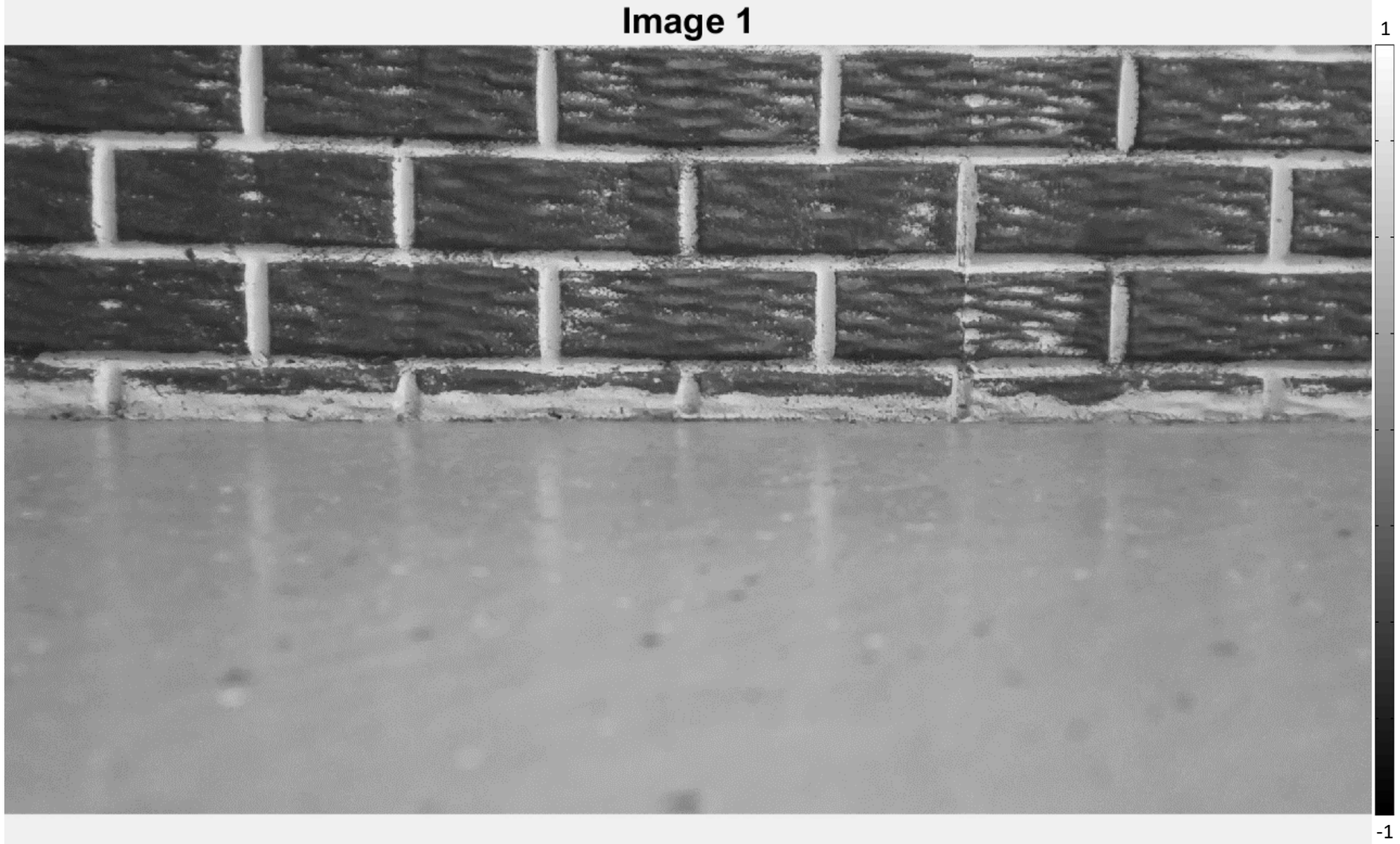
$\sum o_i p_i$

$$r = \frac{\sum p_i o_i - \frac{1}{n}\left(\sum p_i\right)\left(\sum o_i\right)}{\sqrt{\sum p_i^2 - \frac{1}{n}\left(\sum p_i\right)^2}\sqrt{\sum o_i^2 - \frac{1}{n}\left(\sum o_i\right)^2}}$$

Template Matching via DFT



Template Matching via DFT



Peaks, Valleys, and Ridges

Dr. Daniel B. Rowe
Professor of Computational Statistics
Department of Mathematical and Statistical Sciences
Marquette University

Introduction

A recent Marquette University graduate just landed their first job to develop an image recognition system to place eggs in an egg carton.

Image Critical Points

In the same way that we can find critical points for a function, we can find critical points in an image.

We can calculate all of the necessary discrete derivatives using kernels.

1	0	-1
1	0	-1
1	0	-1

/6

1	1	1
0	0	0
-1	-1	-1

/6

1	-2	1
2	-4	2
1	-2	1

/16

1	2	1
-2	-4	-2
1	2	1

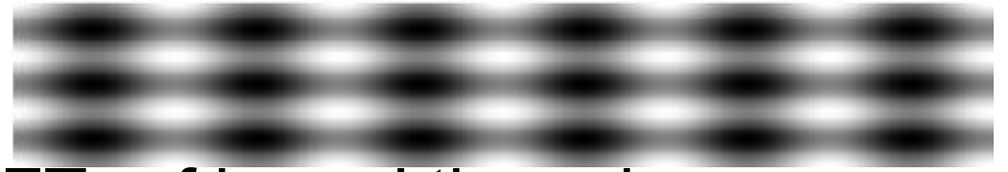
/16

1	0	-1
0	0	0
-1	0	1

/16

$$f_x = \frac{df(x,y)}{dx} \qquad f_y = \frac{df(x,y)}{dy} \qquad f_{xx} = \frac{d^2 f(x,y)}{dx^2} \qquad f_{yy} = \frac{d^2 f(x,y)}{dy^2} \qquad f_{xy} = \frac{d^2 f(x,y)}{dxy}$$

Image Critical Points

$I =$ 

Returning to the graduate's new project, the DFTs of kernel times image

Derivative images using kernels.

$\mathcal{F}\{K\}\mathcal{F}\{I\}$

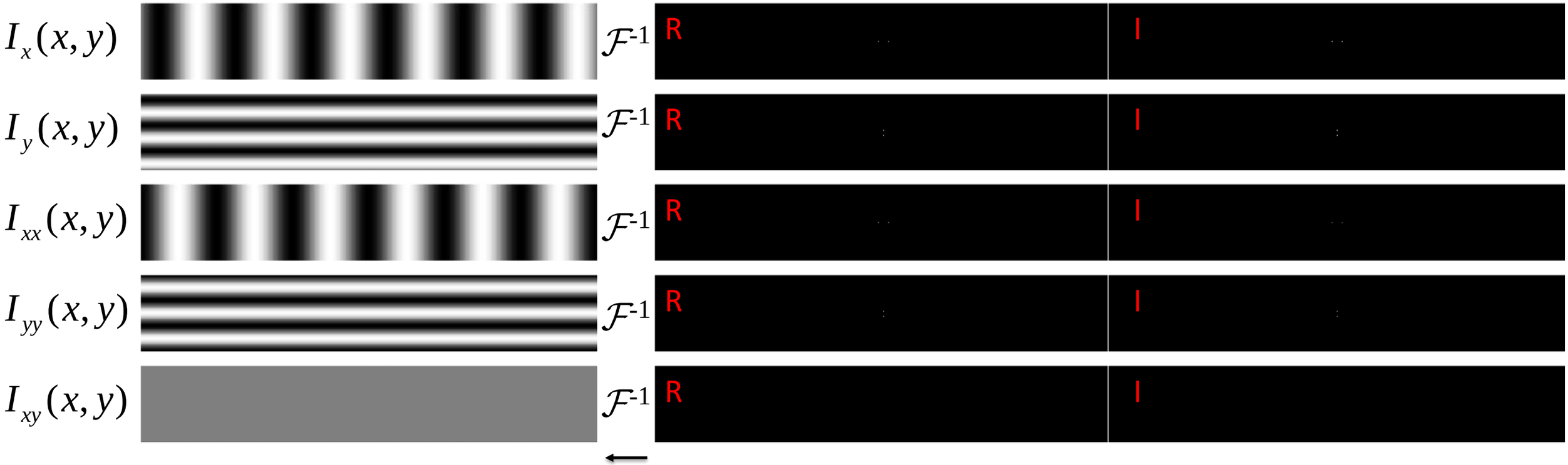
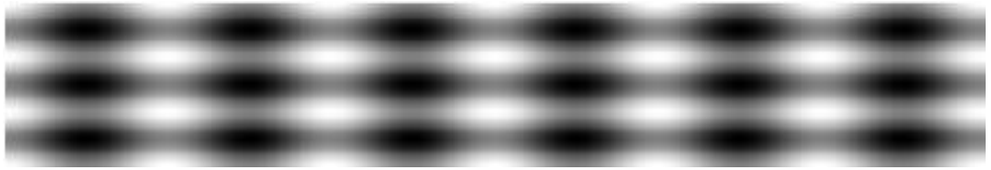


Image Critical Points

$I =$ 

Continuing with the graduate's new project,

Derivative images using kernels.

$I_x(x,y)$ 

$I_y(x,y)$ 

$I_{xx}(x,y)$ 

$I_{yy}(x,y)$ 

$I_{xy}(x,y)$ 

$I_D(x,y)$ 

compare

Derivative images using Calculus.

$f_x(x,y)$ 

$f_y(x,y)$ 

$f_{xx}(x,y)$ 

$f_{yy}(x,y)$ 

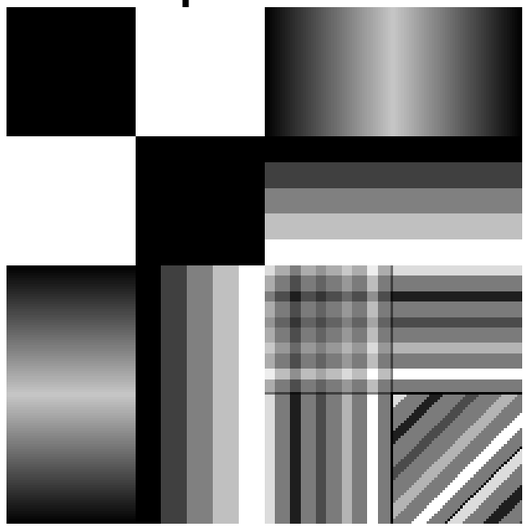
$f_{xy}(x,y)$ 

$f_D(x,y)$ 

Image Critical Points

This process can be applied to other images

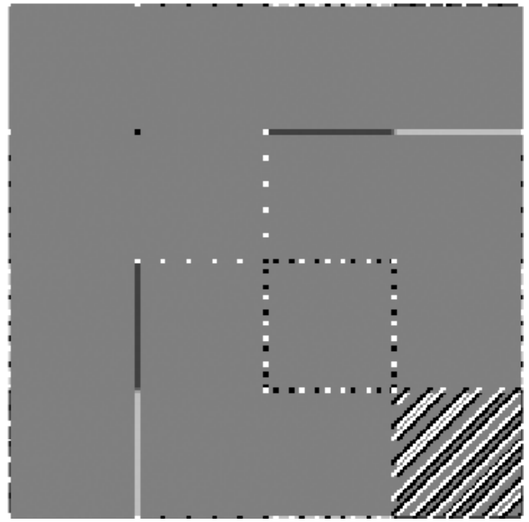
I



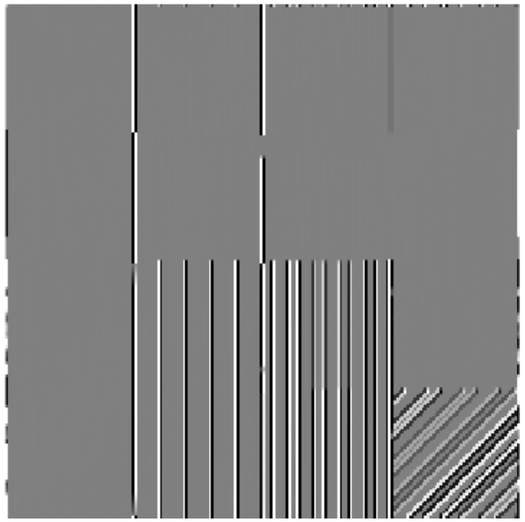
I_x

I_y

I_{xy}



I_{xx}



I_{yy}

I_D

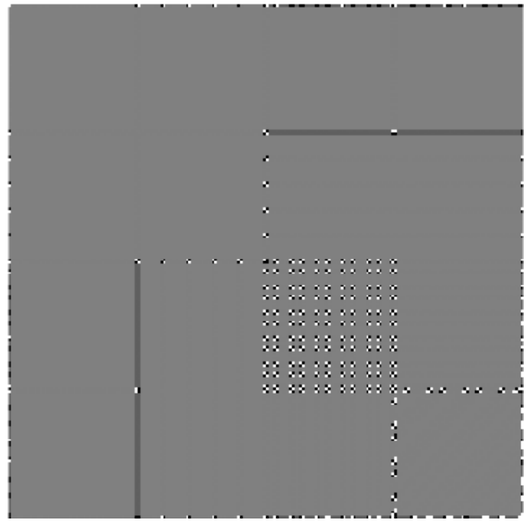
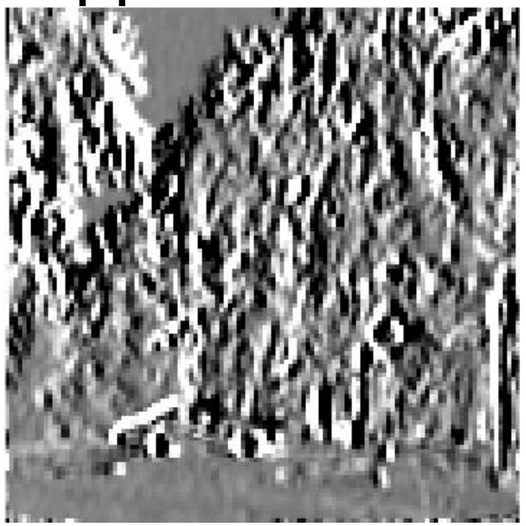


Image Critical Points

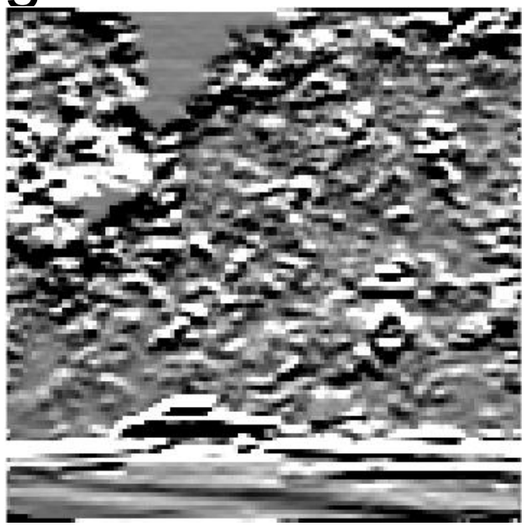
This process can be applied to other images

I

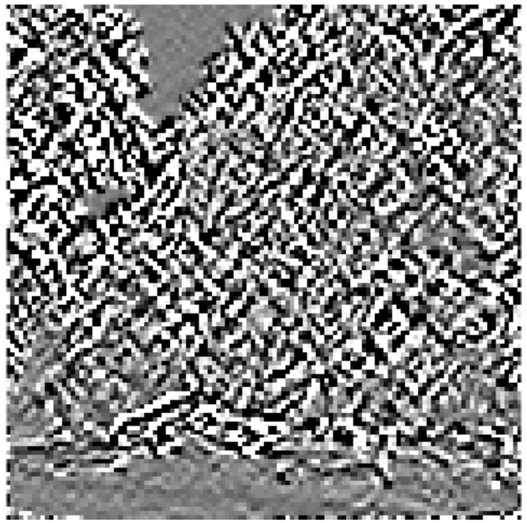
I_x



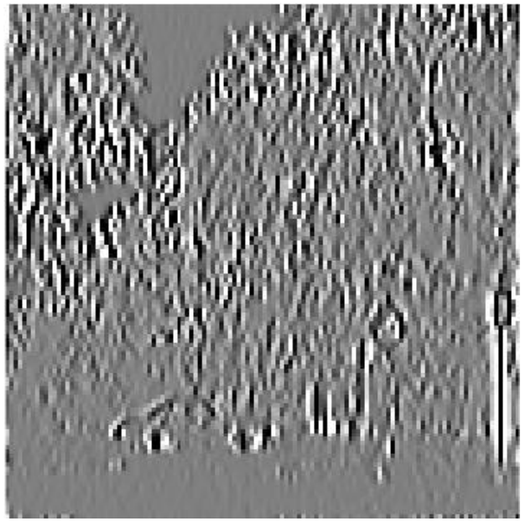
I_y



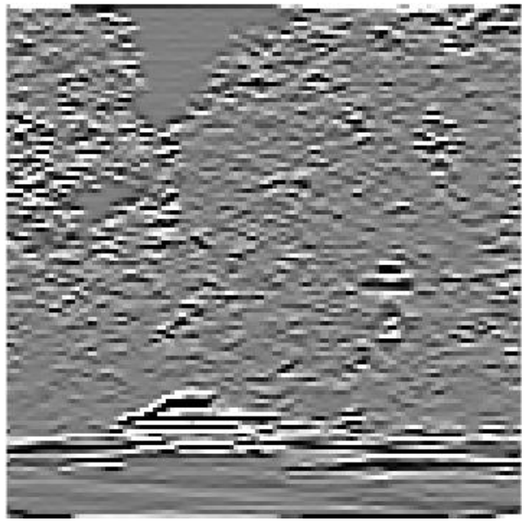
I_{xy}



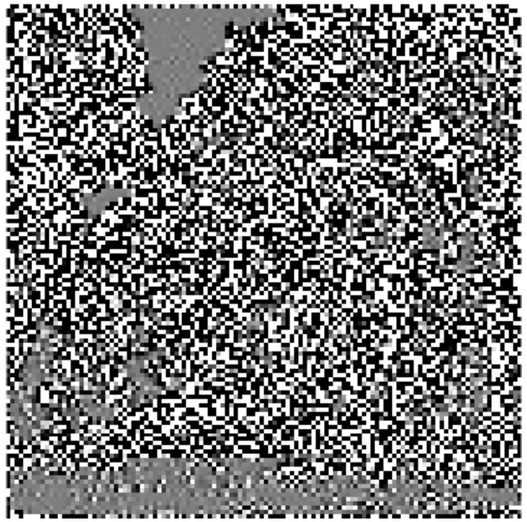
I_{xx}



I_{yy}



I_D



Discussion

We've taken a long journey through many topics.

We are able to load and process images in Matlab.

Convolution via the DFT can accomplish: image smoothing
image sharpening
image frequency decomposition
image template matching
image cartography

In this course you have learned a foundation for a lifetime of imaging.

Discussion

Thank You!