Within Image Processing

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

Outline

Convolution Kernel

Image Smoothing

Image Sharpening

Image NonLinear Filters

Discussion

Homework

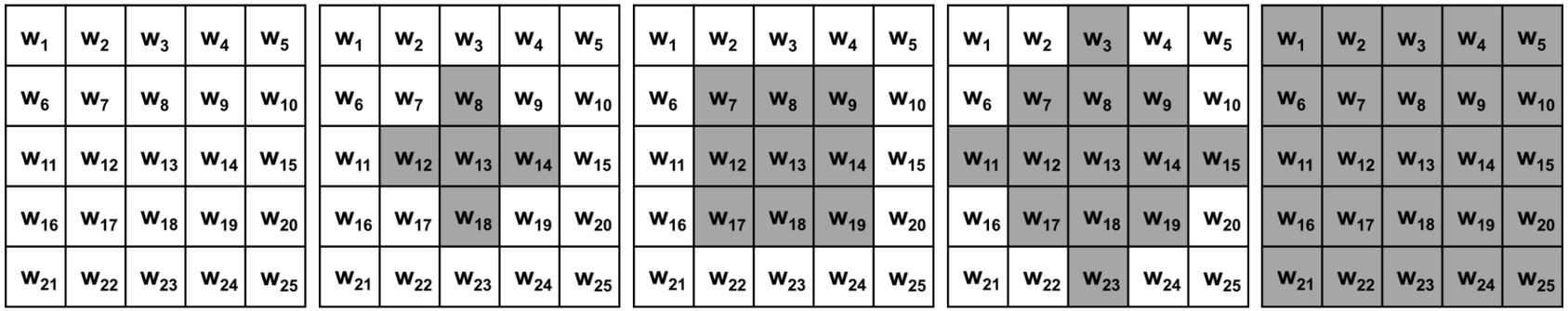
Convolution Kernel

Weights

w

3x3

There are different sizes and weightings to perform different functions.



weights 4 neighbor 8 neighbor 12 neighbor 24 neighbor

Larger more homogenous images use larger kernels.

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 →

$$\begin{matrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{matrix}$$

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.

Smoothing Kernel

Extract
Car Image

Smoothing Kernel

Pixel Values

p

38	37	31	32	28	26	25	27	24	13	35	126	174	194	200	200	200	199	197	194	180	132	94	45	35	41	43	62	39	33	50	70	42	45	44	43	45	38
33	29	30	33	32	26	28	21	16	86	157	171	186	168	170	181	193	196	188	162	161	141	77	66	53	38	43	51	39	40	61	74	60	59	54	47	32	26
35	27	36	36	31	25	14	43	127	172	165	143	139	92	91	99	131	119	92	83	96	123	122	66	72	55	36	49	38	48	61	68	53	47	40	40	24	32
30	26	23	15	26	56	116	174	183	147	122	129	106	121	126	136	157	149	138	135	137	145	162	126	82	95	88	84	73	66	68	63	46	32	31	40	27	27
21	38	98	152	178	179	173	180	184	182	179	185	190	189	180	182	180	185	186	182	192	197	197	198	198	198	197	197	197	193	189	183	169	154	141	127	84	50
60	135	144	124	114	109	101	90	58	59	116	123	105	112	114	116	133	147	128	92	116	122	101	84	83	98	101	105	133	139	136	130	133	132	125	91	68	60
89	95	94	98	97	97	93	94	124	142	139	124	123	111	121	123	122	109	131	172	177	150	133	125	128	131	111	79	75	69	72	77	82	72	73	87	58	63
124	41	47	49	49	50	28	16	31	36	31	45	51	51	49	49	52	47	31	22	27	40	46	46	40	58	40	19	55	64	44	29	57	83	135	123	45	43
93	34	38	40	40	38	28	60	104	95	45	31	45	44	43	44	44	44	46	45	45	46	45	44	44	45	34	66	99	113	74	37	29	35	84	100	51	43
105	84	69	61	45	42	37	71	90	102	68	31	40	40	39	38	40	40	42	40	39	39	40	40	39	38	32	83	99	113	90	74	22	24	56	69	78	92
119	122	120	123	126	127	108	107	175	159	61	18	24	17	39	46	50	45	47	51	56	51	32	30	29	27	43	88	134	170	140	84	15	46	135	154	178	169
146	137	137	139	99	99	89	57	94	86	27	9	21	75	121	122	124	122	122	121	128	135	135	133	124	117	99	68	96	119	74	17	15	70	94	91	100	107

Image Smoothing

Weights w 

$$\sum_{i=1}^{9} w_i = 1$$

Image Smoothing

Weights

w

$\frac{1}{5}$

0	1	0
1	1	1
0	1	0

Pixel Values

p

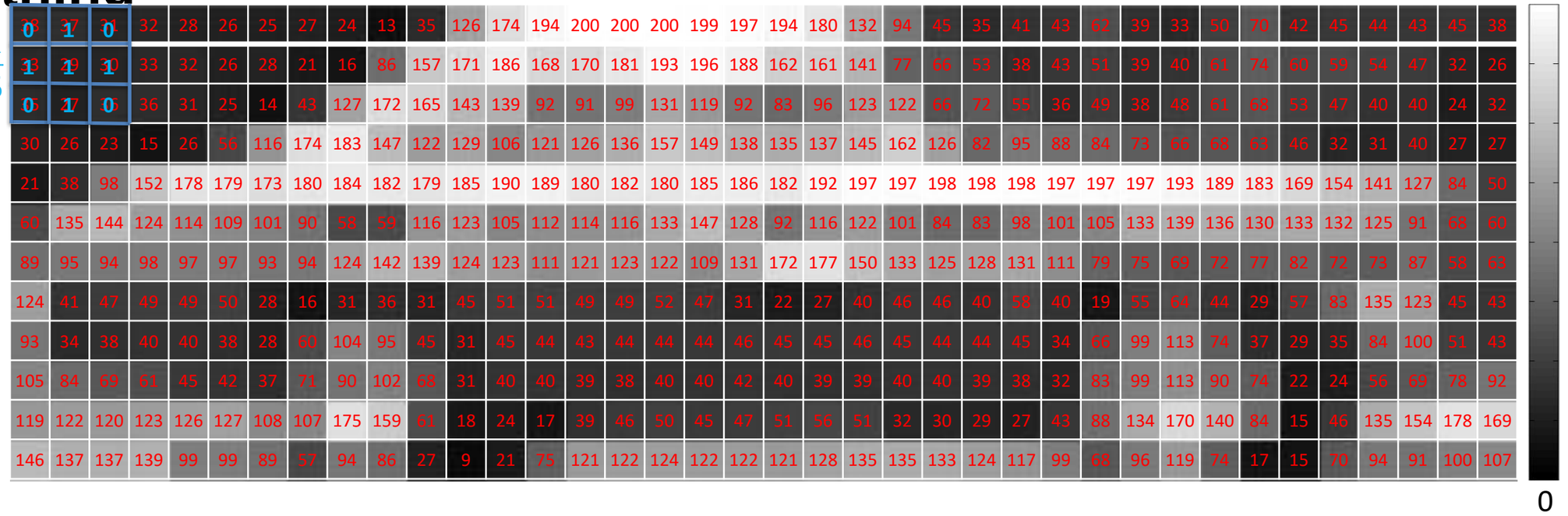
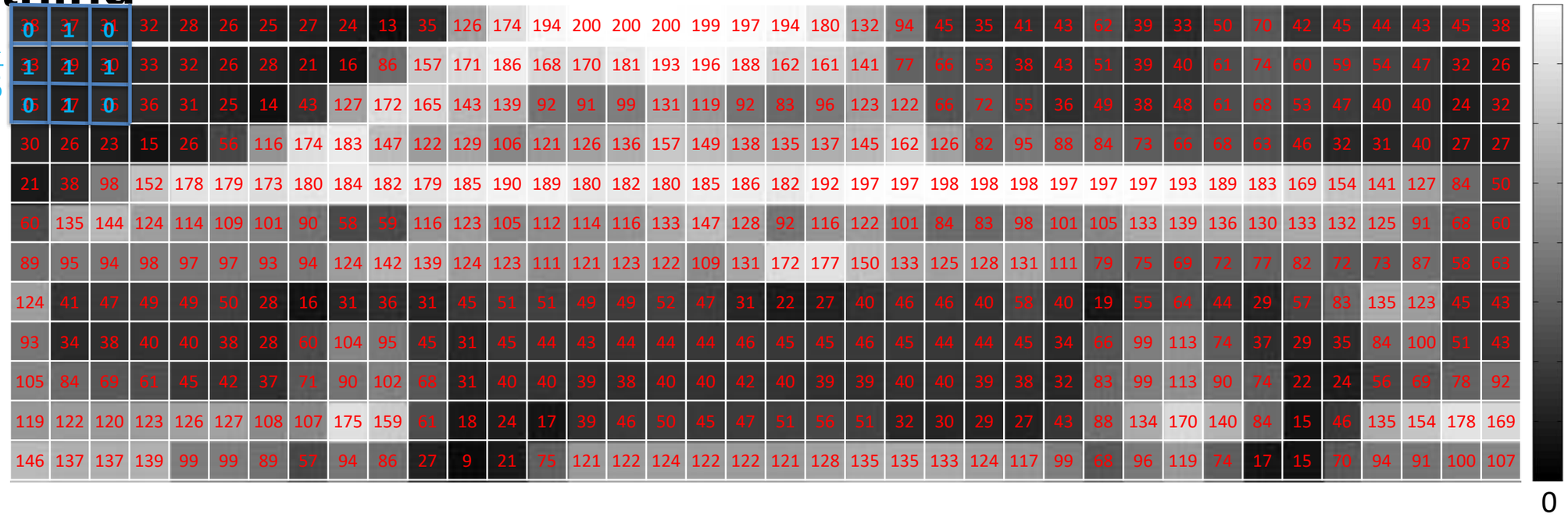


Image Smoothing

Weights w $\frac{1}{5}$

Pixel Values p



New image

$$q_5 = .2(37) + .2(33) + .2(29) + .2(30) + .2(27) \equiv 31$$

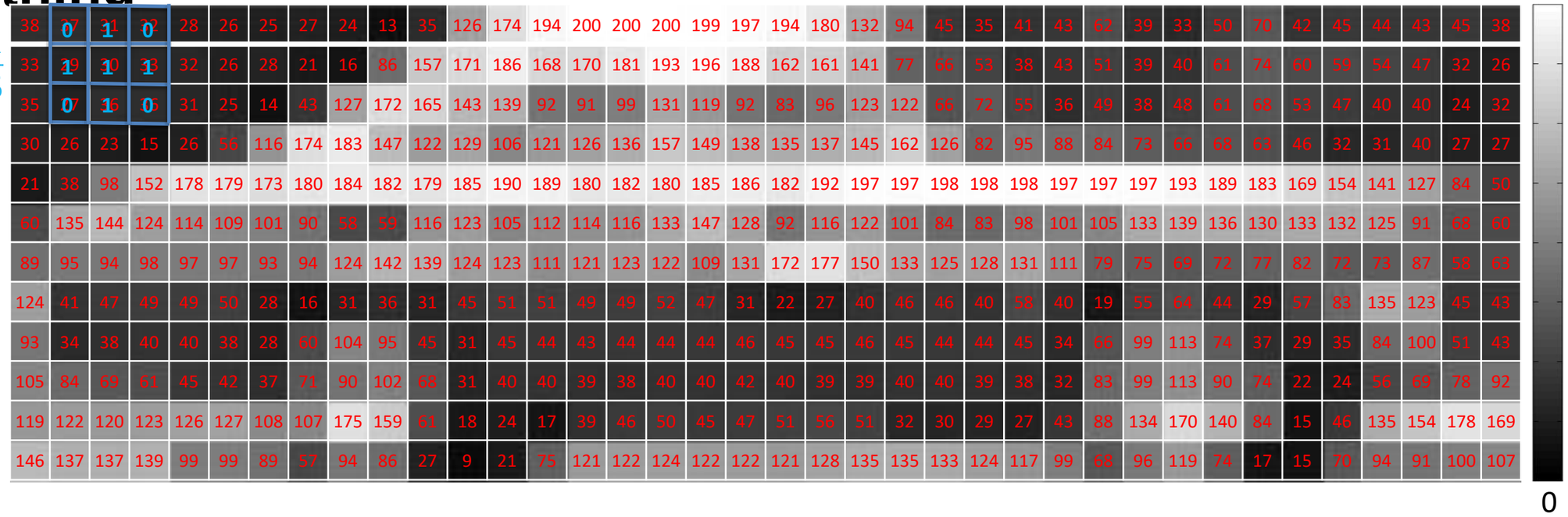
$$\begin{bmatrix} & & \\ & q_5 & \\ & & \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} \times \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 \end{bmatrix}$$

$$q_5 = \sum_{i=1}^{9} w_i p_i$$

Image Smoothing

Weights w → $\frac{1}{5}$

Pixel Values p →



New image

$$q_5 = .2(31) + .2(29) + .2(30) + .2(32) + .2(36) \equiv 32$$

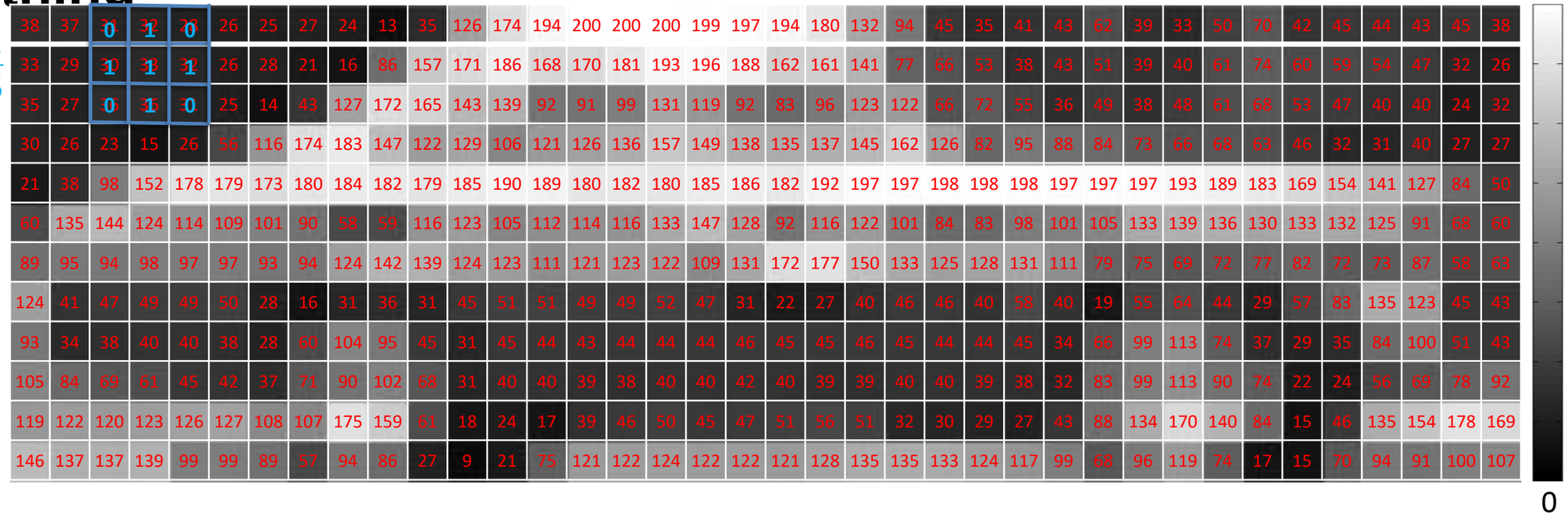
$$\begin{bmatrix} & & \\ & q_5 & \\ & & \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} .\times \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 \end{bmatrix}$$

$$q_5 = \sum_{i=1}^{9} w_i p_i$$

Image Smoothing

Weights w → $\frac{1}{5}$

Pixel Values p →



New image

$$q_5 = .2(32) + .2(30) + .2(33) + .2(32) + .2(36) \equiv 33$$

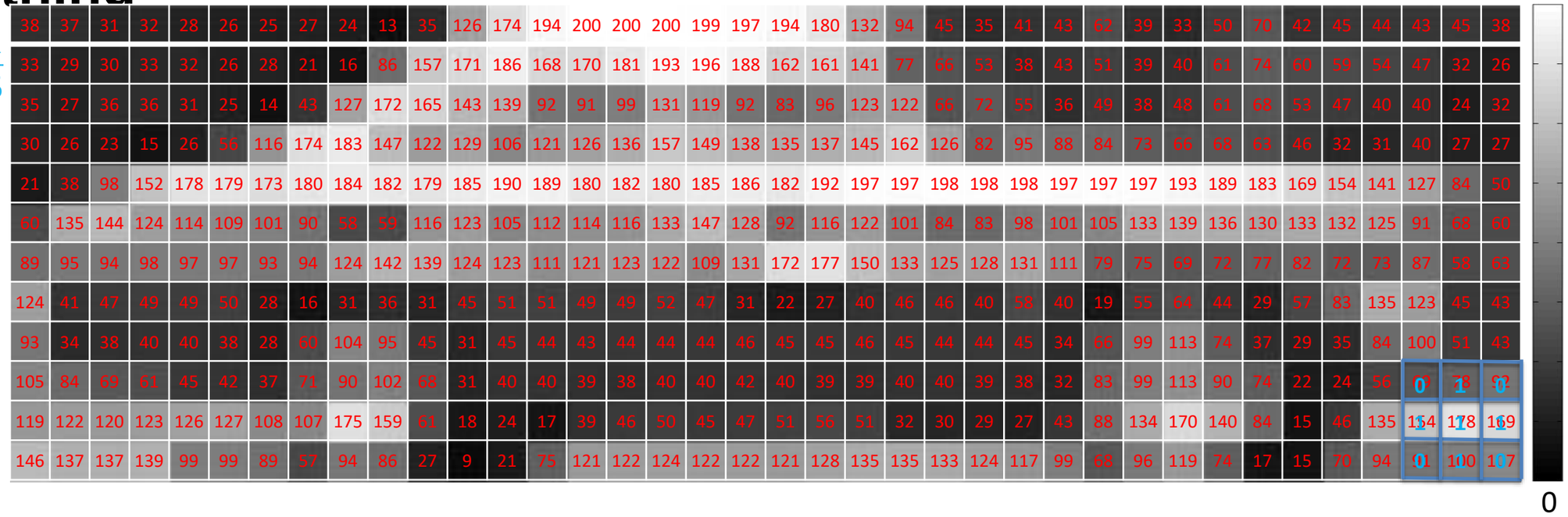
$$\begin{bmatrix} & & \\ & q_5 & \\ & & \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} \times \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 \end{bmatrix}$$

$$q_5 = \sum_{i=1}^{9} w_i p_i$$

Image Smoothing

Weights w → $\frac{1}{5}$

Pixel Values p →



New image

$$q_5 = .2(78) + .2(154) + .2(178) + .2(169) + .2(100) \equiv 136$$

$$\begin{bmatrix} & & \\ & q_5 & \\ & & \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} \times \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 \end{bmatrix}$$

$$q_5 = \sum_{i=1}^{9} w_i p_i$$

Image Smoothing

Weights

w

$\frac{1}{5}$

Pixel Values

p

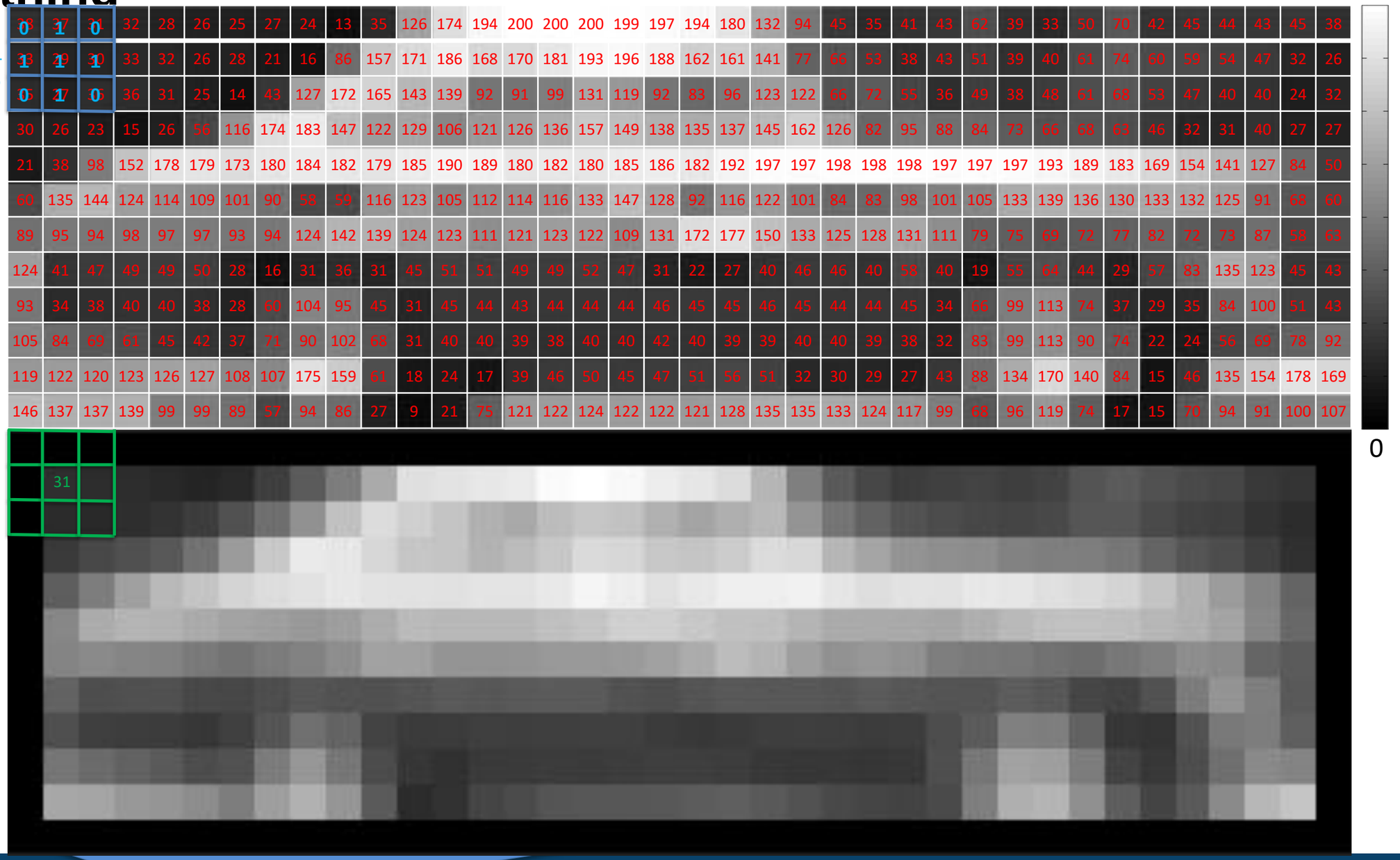


Image Smoothing

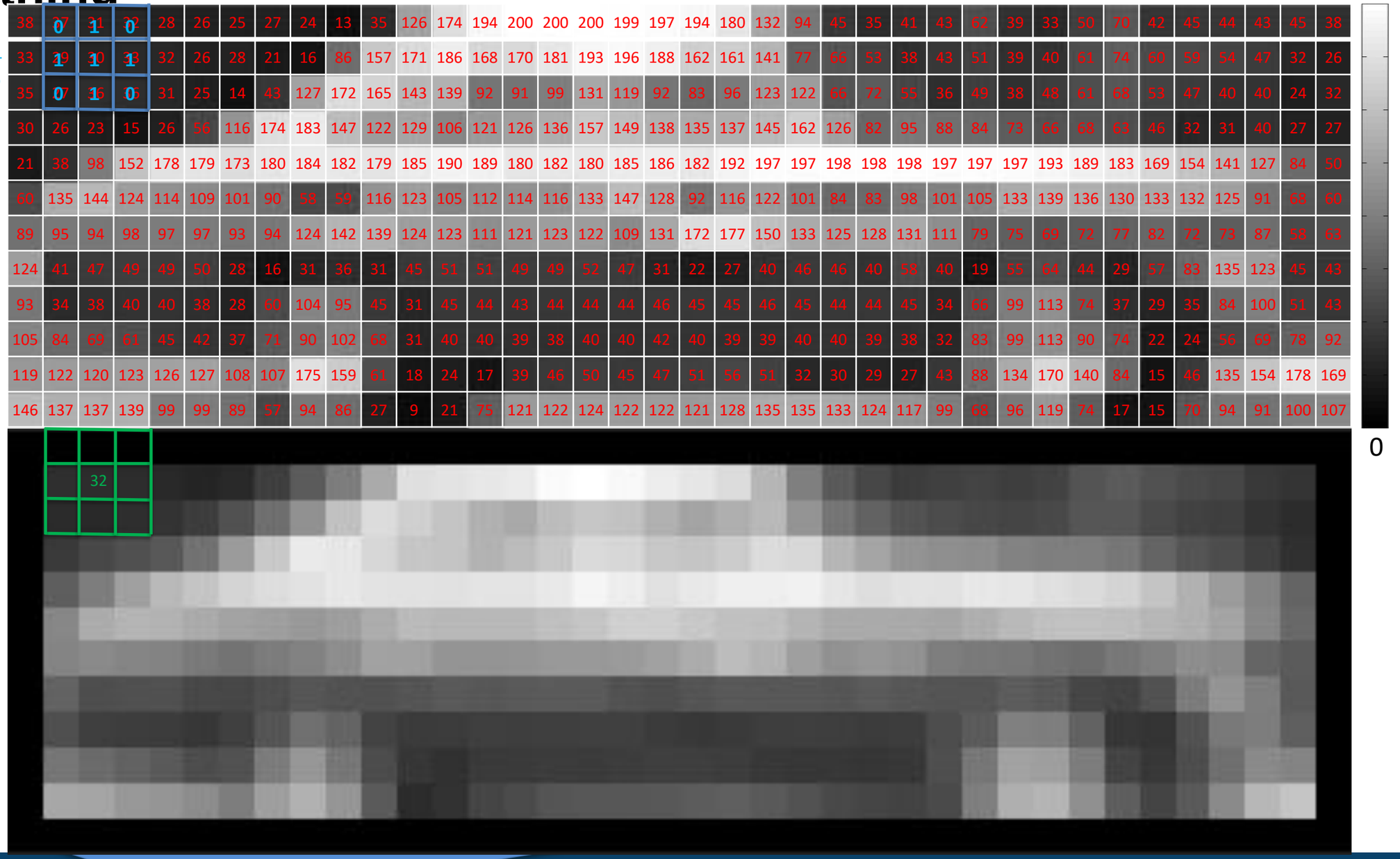
Weights

w

$\frac{1}{5}$

Pixel Values

p



New image

Image Smoothing

Weights

w

$\frac{1}{5}$

Pixel Values

p

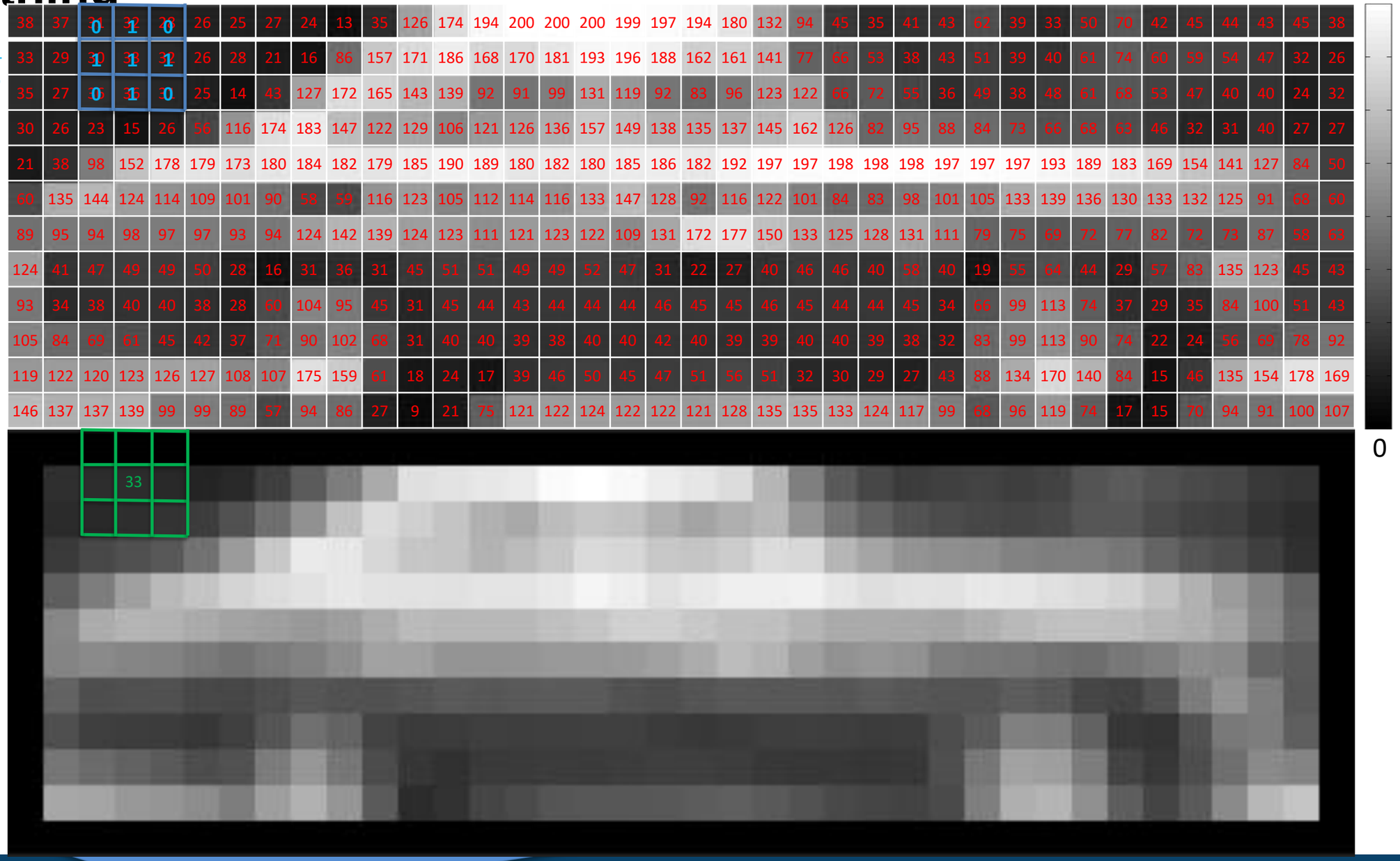
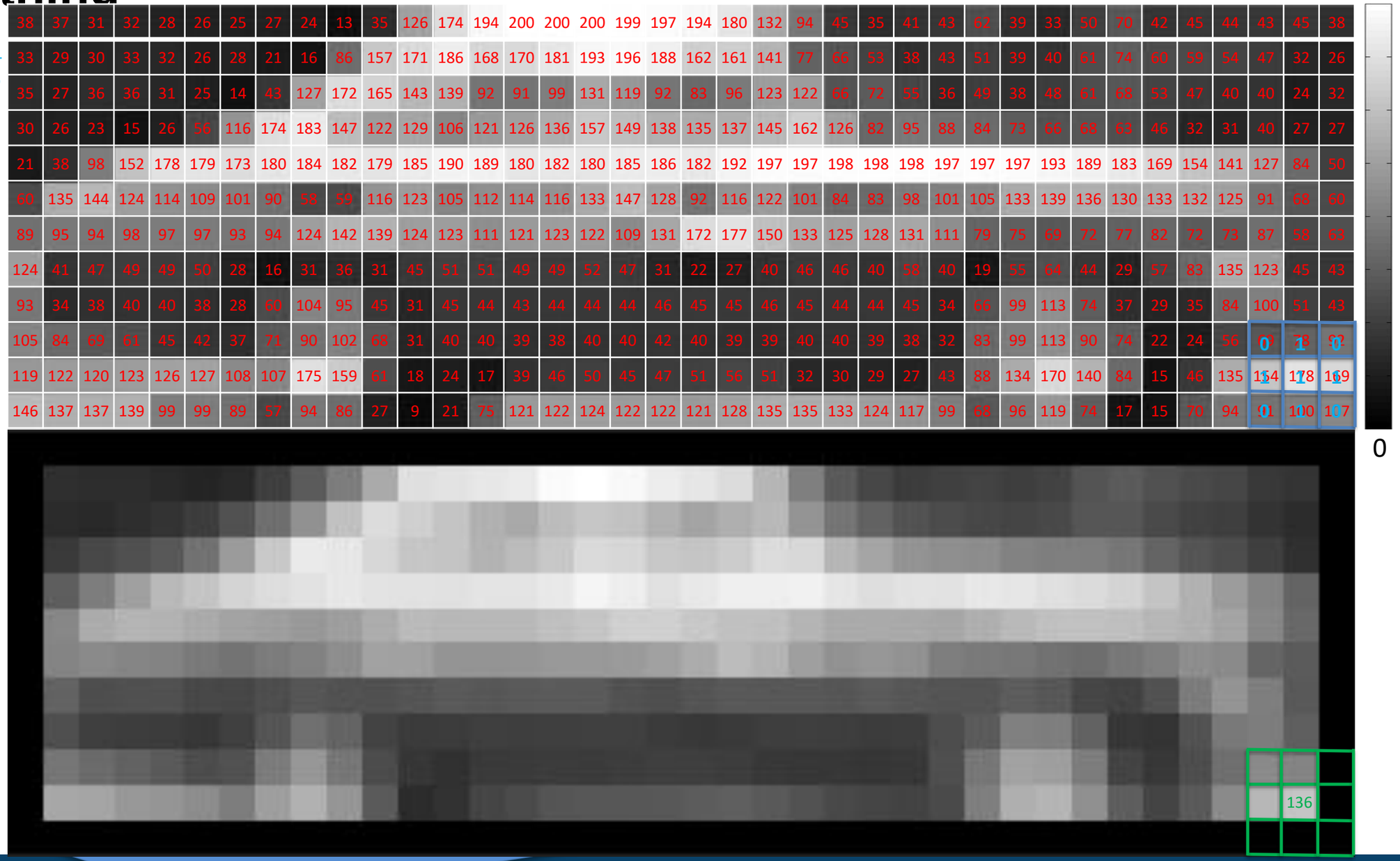


Image Smoothing

Weights w $\frac{1}{5}$

Pixel Values p



New image processed

Image Smoot

Weights

w

Pixel Values

p

New image processed

$\frac{1}{5}$

0	1	0
1	1	1
0	1	0

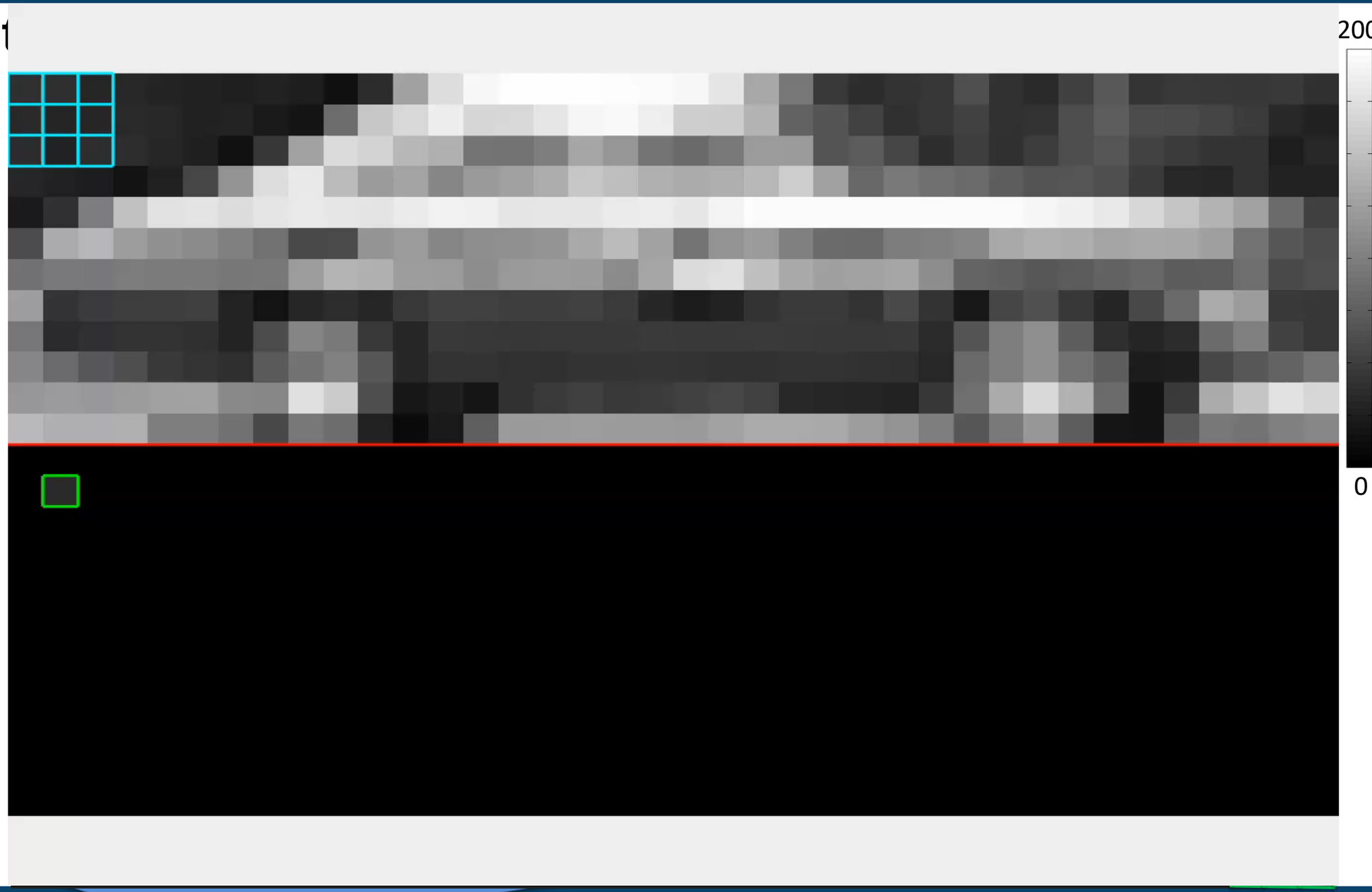


Image Smoothing

```
load cardata.txt
[n,p]=size(cardata);
nx=sqrt(n);, ny=nx;
fxy=reshape(cardata,[ny,nx])';
imwrite(uint8(fxy),'CarImage.tif','tif',...
'Resolution',[300 300],'Compression','none');

mx=200;

figure;
imagesc(fxy,[0,mx])
axis image, colormap(gray), axis off

car=fxy(93:104,27:64);
figure;
imagesc(uint8(car),[0,mx])
axis image, colormap(gray), axis off
```

```
[n,m]=size(car);
carSm=zeros(n,m);
for j=1+1:n-1
    for i=1+1:m-1
        carSm(j,i)=(car(j-1,i)+car(j,i-1)...
        +car(j,i)+car(j,i+1)+car(j+1,i+1))/5;
    end
end

figure;
imagesc(carSm,[0,mx])
axis image, colormap(gray), axis off
```

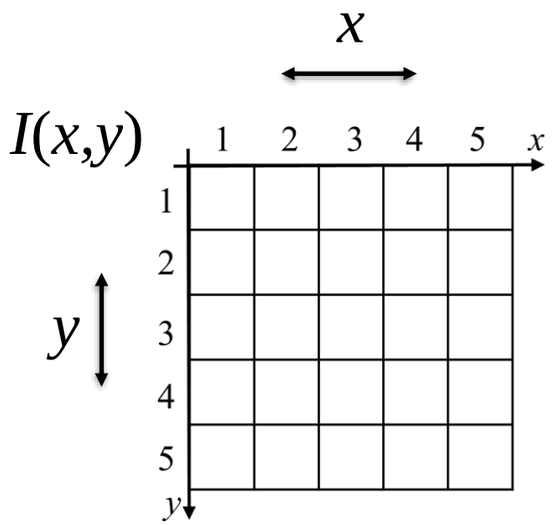
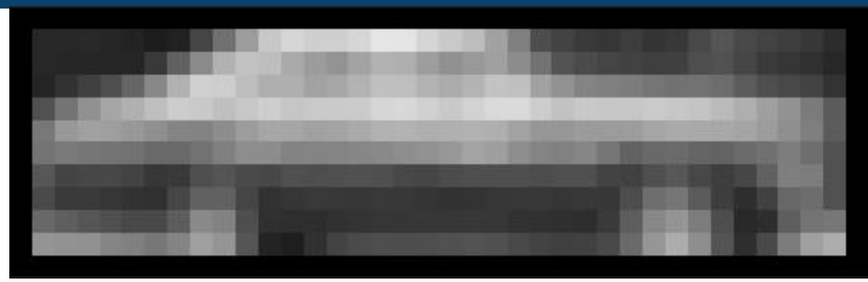


Image Smoothing



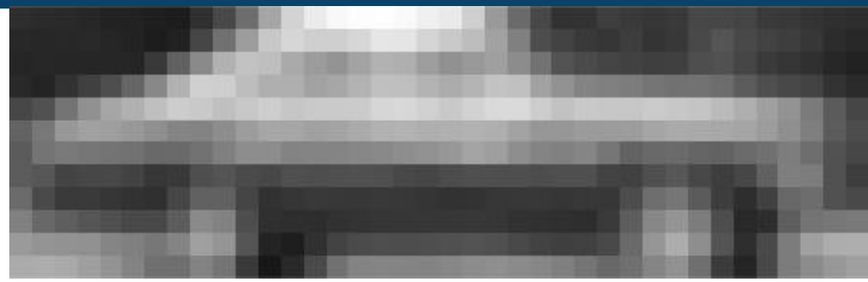
As you can see, we did not process the edges.

If we had placed the center of the kernel along an edge, we would be missing some values in the image to use as pixel values to weight into the new image.

There are a few options that we can pursue.

Image Smoothing

1) Reweight $\frac{1}{3}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$



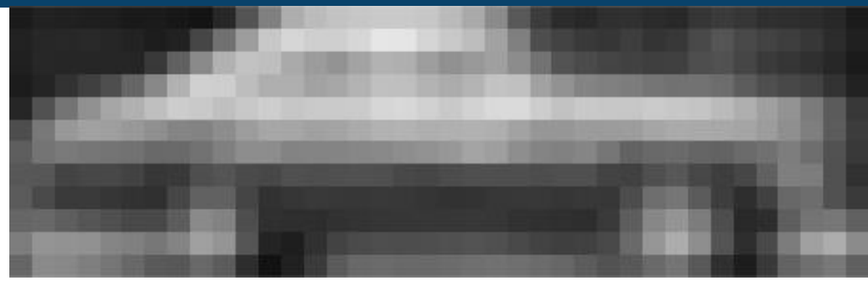
A good option.

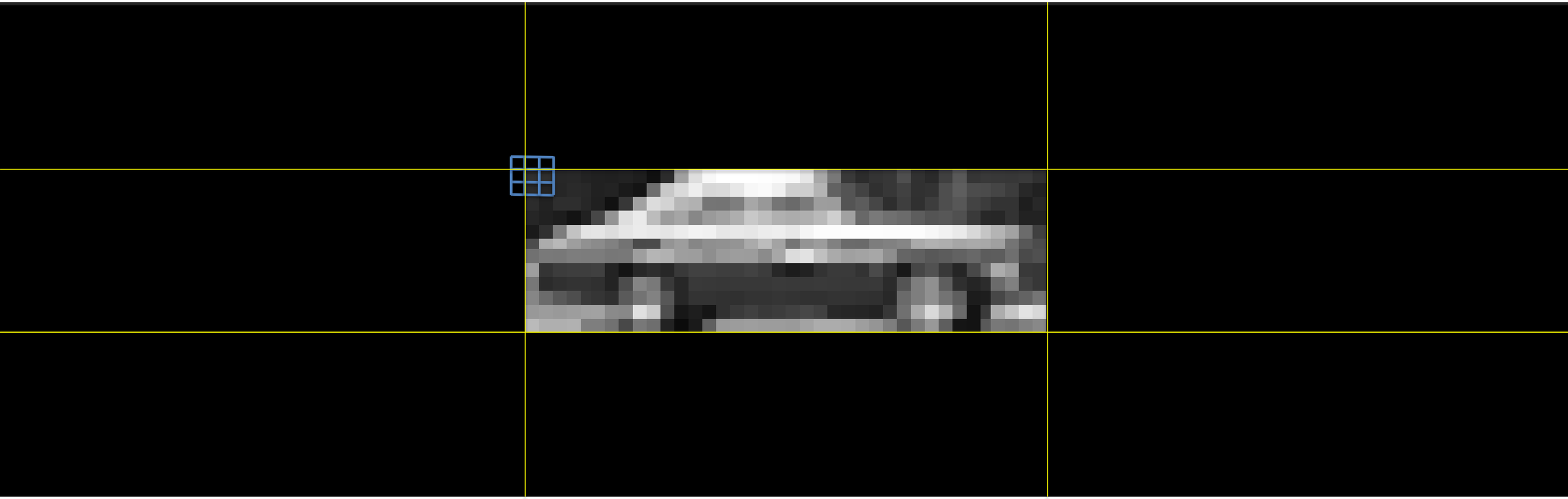
Image Smoothing

$\frac{1}{5}$

0	1	0
1	1	1
0	1	0

2) Zero Pad

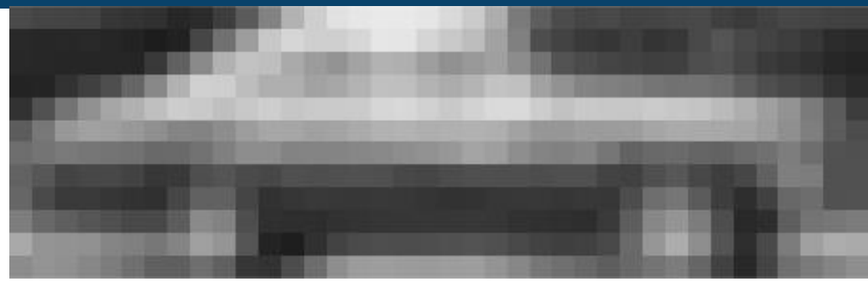




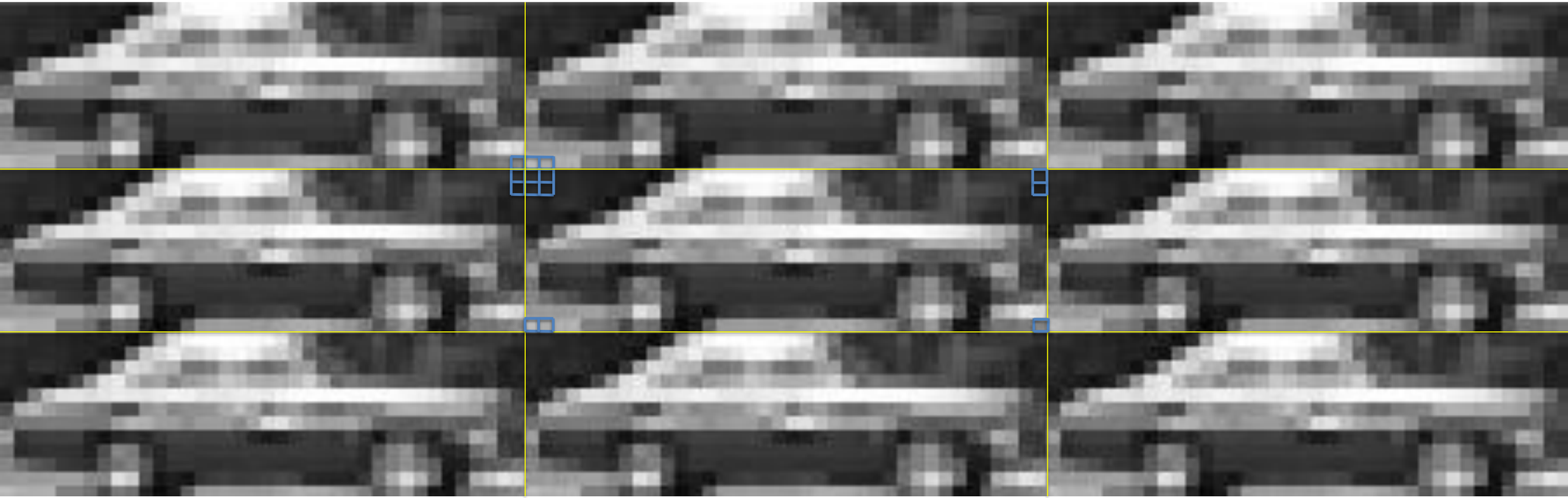
The pixels come up short.

Image Smoothing

$$\frac{1}{5}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

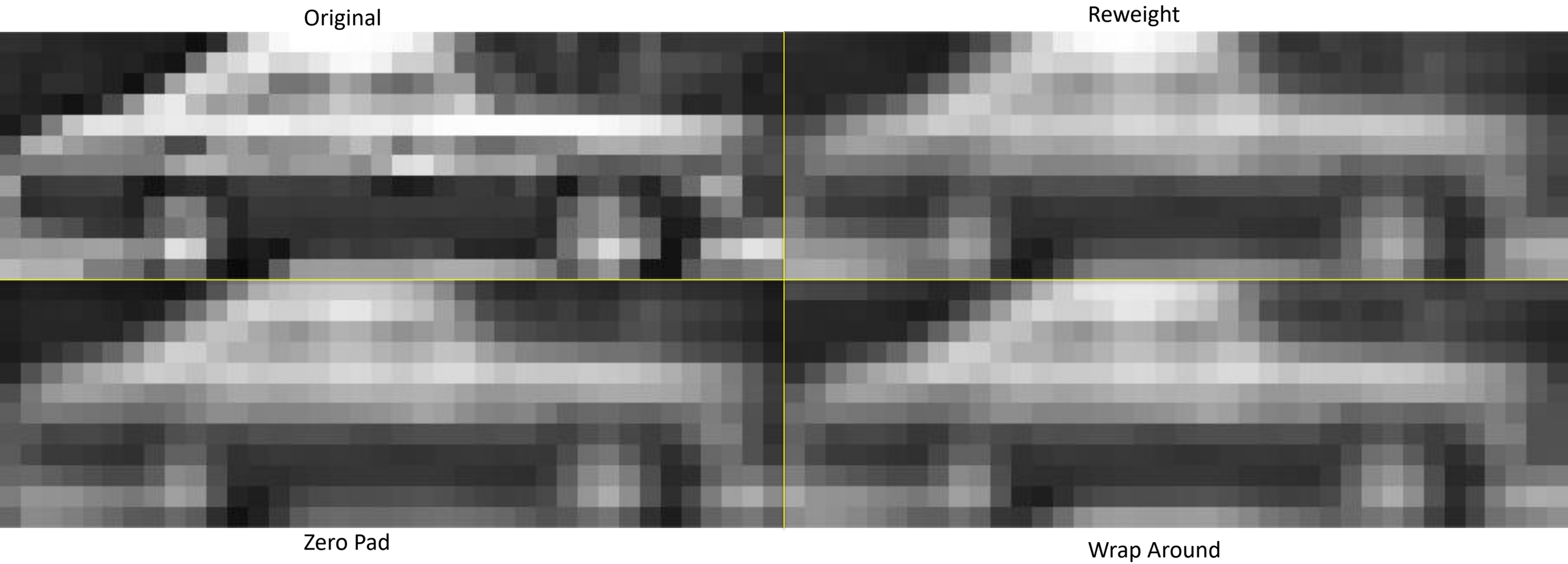


3) Wrap



This is what we will eventually do via DFT.

Image Smoothing

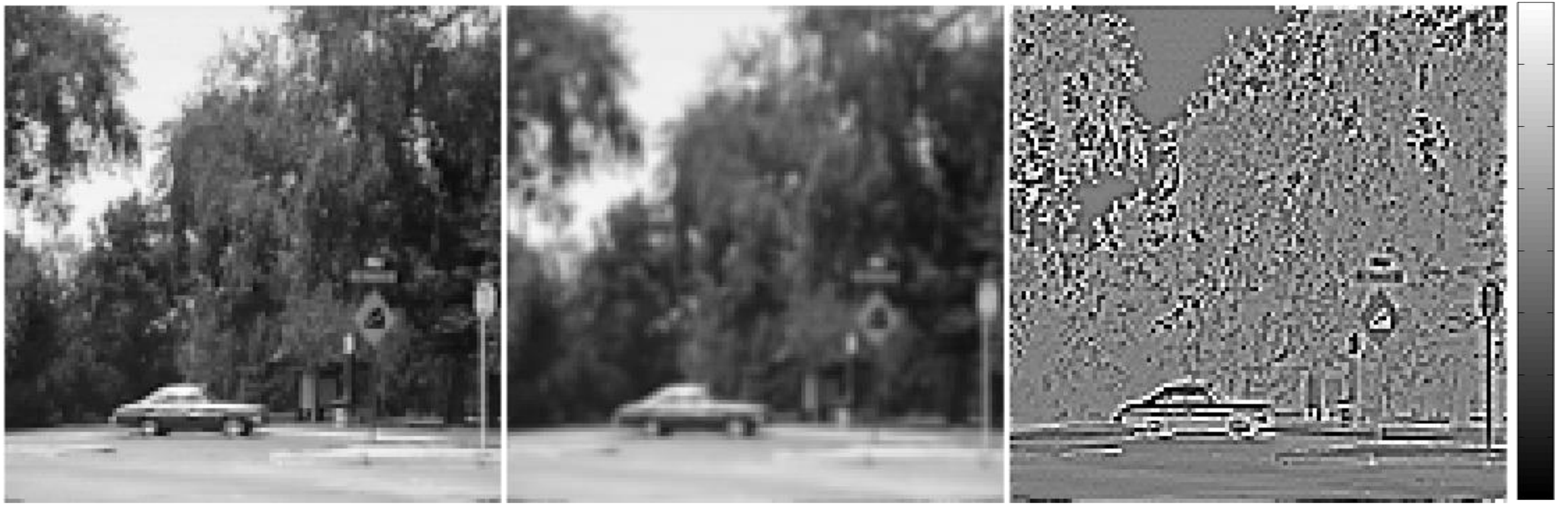


For large images edges don’t matter when the focus is the center.

Image Smoothing

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

Apply to whole image and examine the difference.



Original

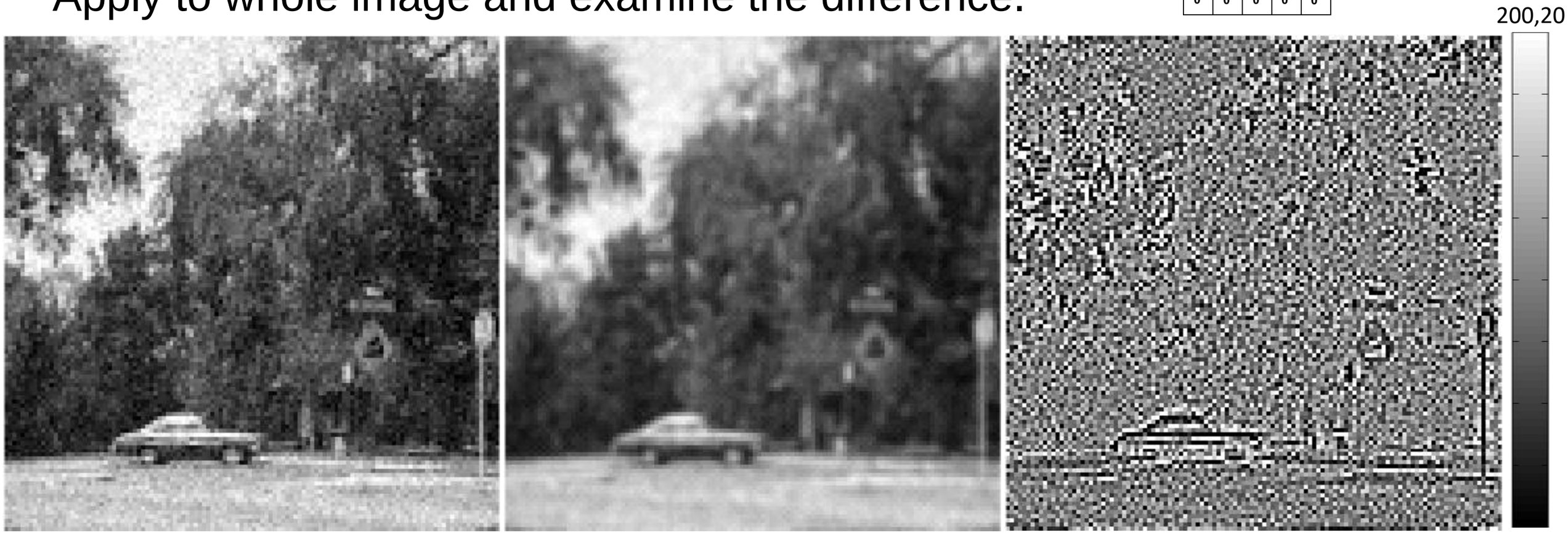
Smoothed

Difference

Image Smoothing

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

Apply to whole image and examine the difference.



Original
Gaussian

Smoothed

Difference

Image Smoothing

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

Apply to whole image and examine the difference.

Original
Salt-N-Pepper

Smoothed

Difference

Image Sharpening

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

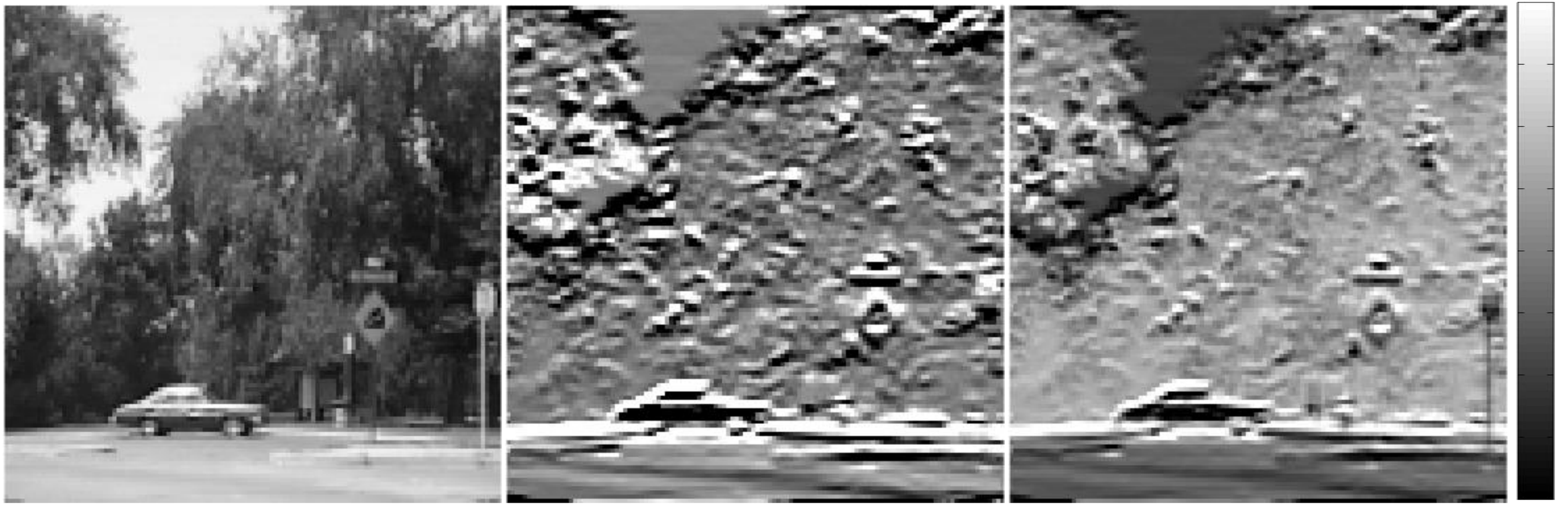
Original

Laplace

Difference

Image Sharpening

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



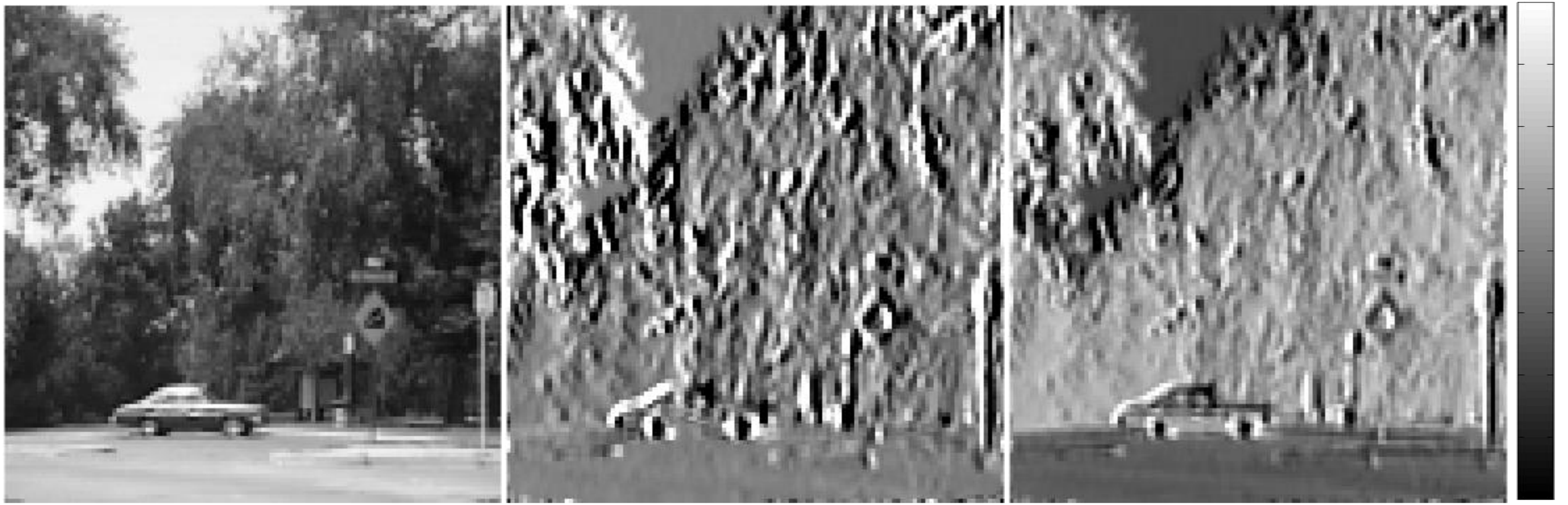
Original

U-D Gradient

Difference

Image Sharpening

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

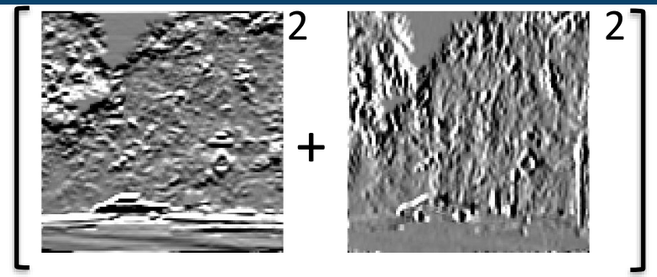


Original

L-R Gradient

Difference

Image Sharpening



Original

Magnitude Gradient

Difference

Image NonLinear Filters

So far we have talked about linear filters where the new pixel value q is a linear combination of the neighborhood pixels p in the original image.

We can use not linear functions of the neighborhood pixels.
Median, minimum, maximum, sum of squares, exponential, …

Or more complicated neighborhoods.

Image NonLinear Filters

MEDIAN

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

Apply to whole image and examine the difference

Original

Median 8

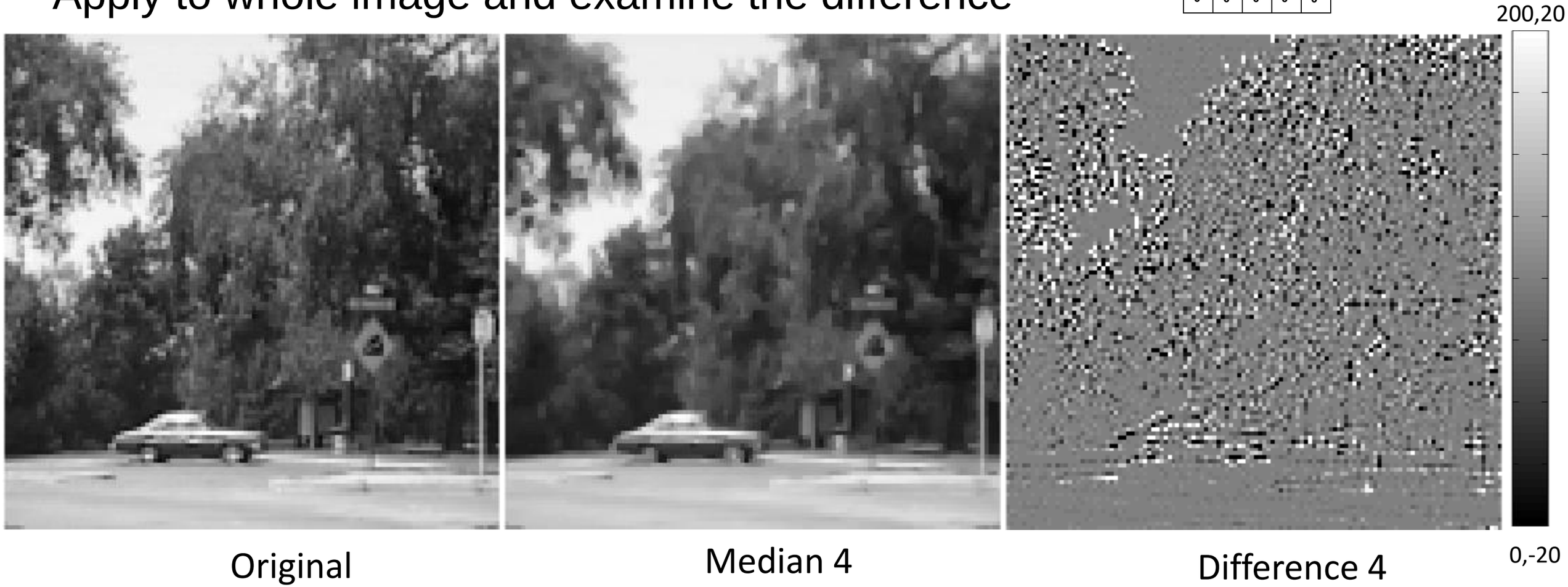
Difference 8

Image NonLinear Filters

MEDIAN

0	0	0	0	0
0	0	1	0	0
0	1	1	1	0
0	0	1	0	0
0	0	0	0	0

Apply to whole image and examine the difference



Original

Median 4

Difference 4

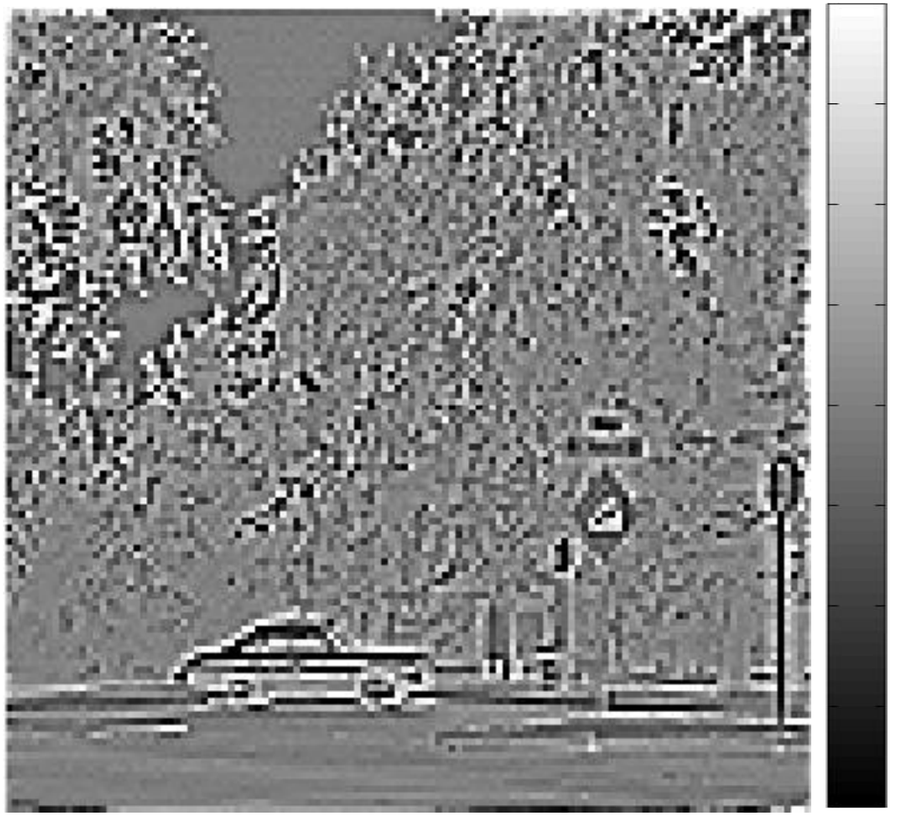
Image NonLinear Filters

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

Apply to whole image and examine the difference

Original

Smoothed



Difference

Image NonLinear Filters

MEDIAN

0	0	0	0	0
0	0	1	0	0
0	1	1	1	0
0	0	1	0	0
0	0	0	0	0

Apply to whole image and examine the difference

Original
Salt-N-Pepper

Median 4

Difference 4

Image NonLinear Filters

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

Apply to whole image and examine the difference.

Original
Salt-N-Pepper

Smoothed

Difference

Image NonLinear Filters

An issue is that when we are along an edge. Light gets averaged with dark.

We can have the filter for a pixel dynamically change based upon its neighborhood.

The DAN (directionally averaged neighborhood) filter.

Average of most similar neighborhood.

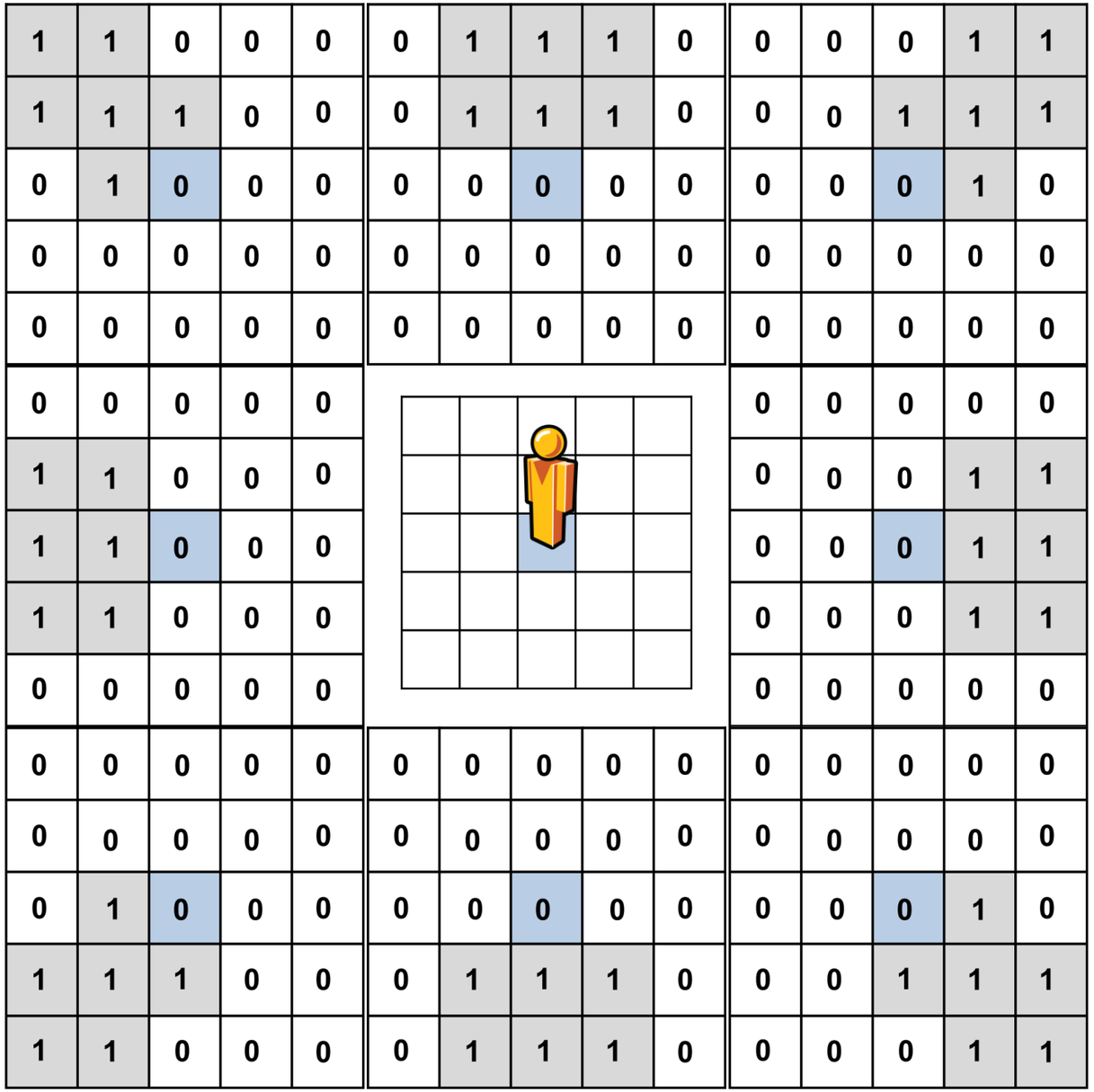
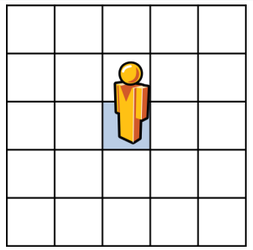


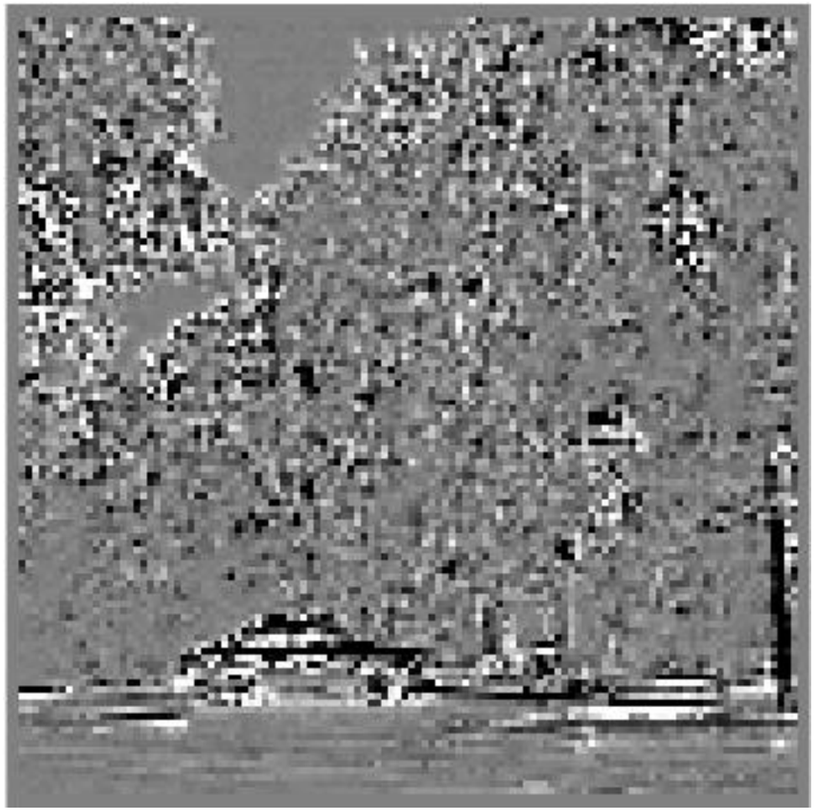
Image NonLinear Filters



Apply to whole image and examine the difference.

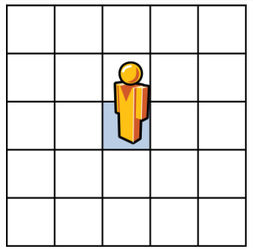
Original

Smoothed DAN



Difference

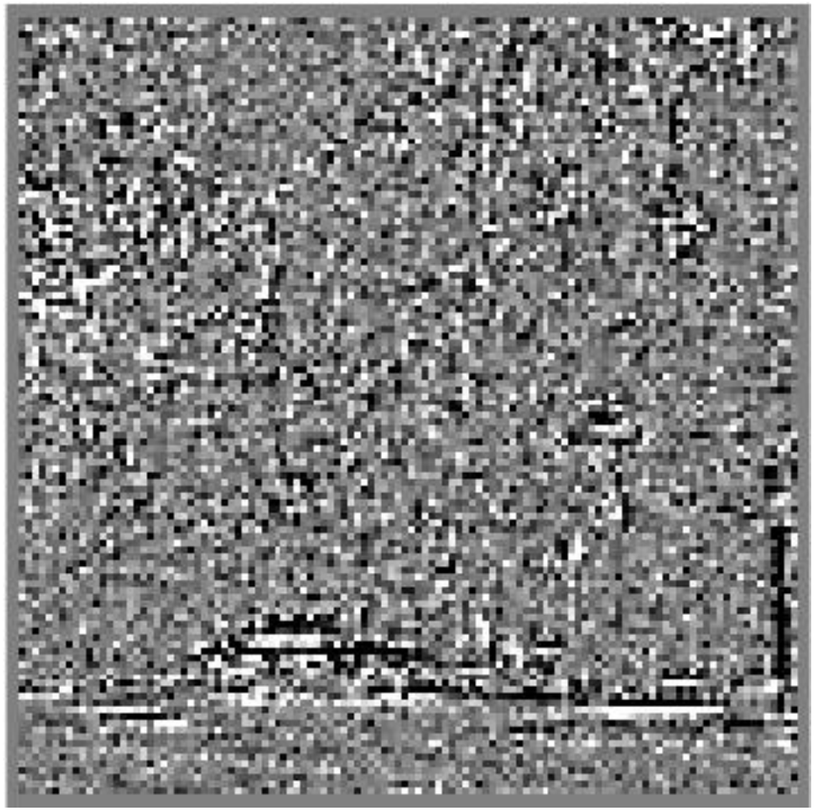
Image NonLinear Filters



Apply to whole image and examine the difference.

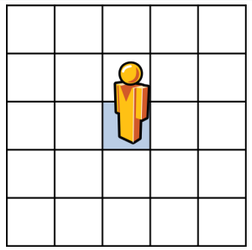
Original
Gaussian

Smoothed DAN



Difference

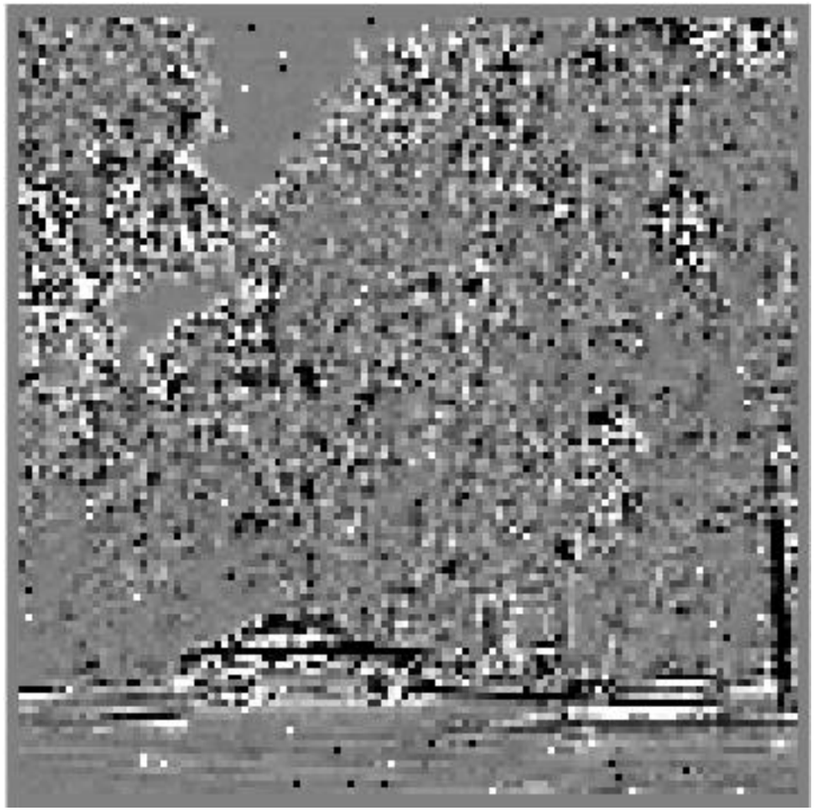
Image NonLinear Filters



Apply to whole image and examine the difference.

Original
Salt-N-Pepper

Smoothed DAN



Difference

Discussion

There are many within image filters that we can apply.

Each has it’s own properties and scenarios when should be applied.

Discussion

Questions?

Homework 2

1. Modify my smoothing code to smooth border pixels with wrap-around. Write YOUR own convolution code, do not use a Matlab function!

2. Apply 3×3 4-neighbor filter to grey Fr. Marquette image using YOUR code.

3. Apply another filter to Father Marquette image using YOUR code.

4*.Make a 7×7 filter and apply it to your own image using YOUR code.

5*.Create your own filter and apply it to your own image using YOUR code.

*For students in MSSC 5770.