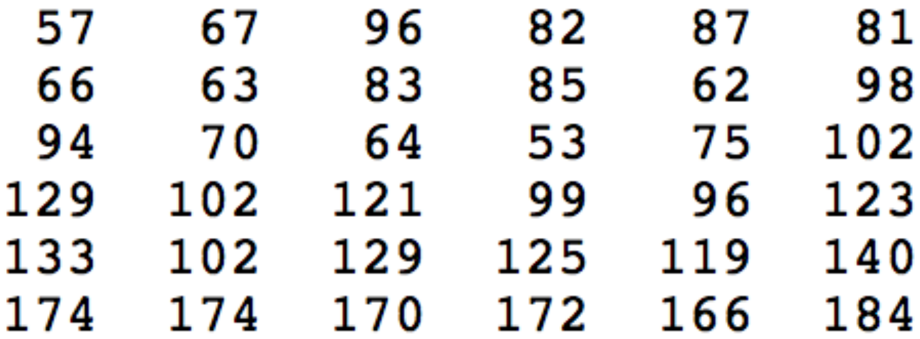
Introduction

Image Processing Lecture 1

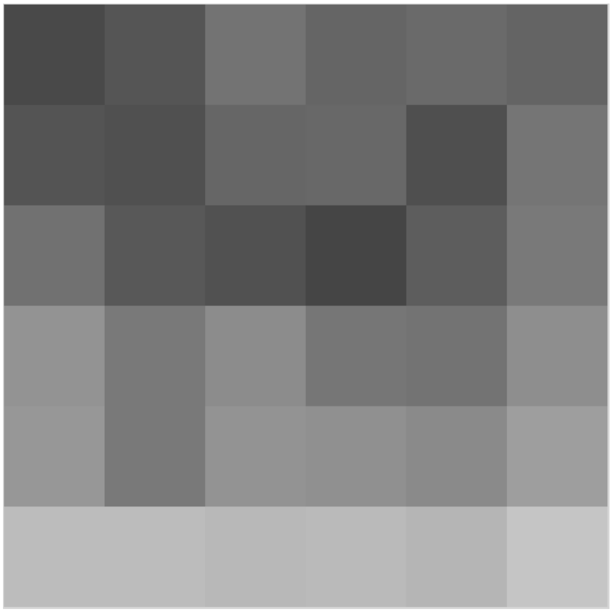
Nicholas Dwork

2D Array

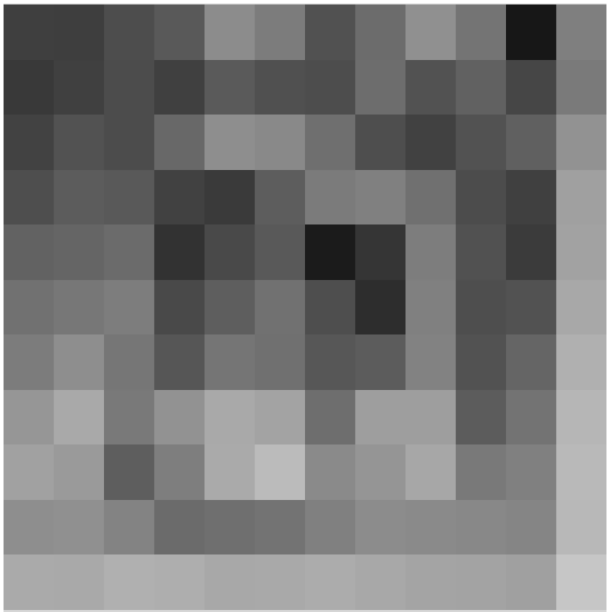


```
 57   67   96   82   87   81
 66   63   83   85   62   98
 94   70   64   53   75  102
129  102  121   99   96  123
133  102  129  125  119  140
174  174  170  172  166  184
```

Rather than showing the numbers, we can show corresponding colors. 0=black, and 255=white.



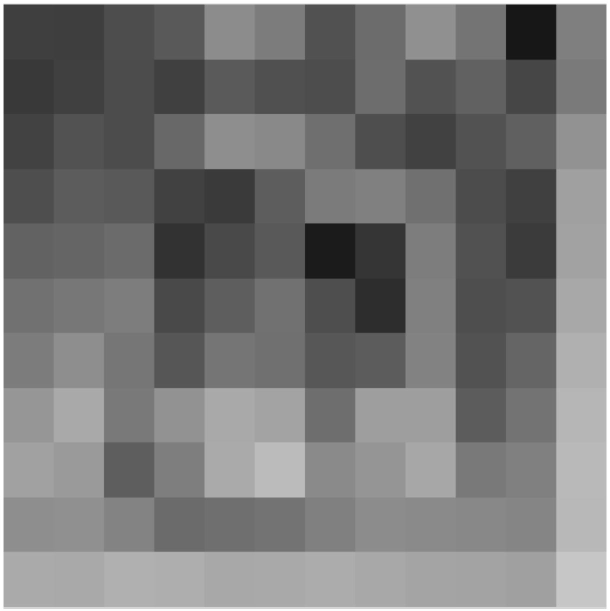
**Here's a larger array.
0=black, and 255=white.**



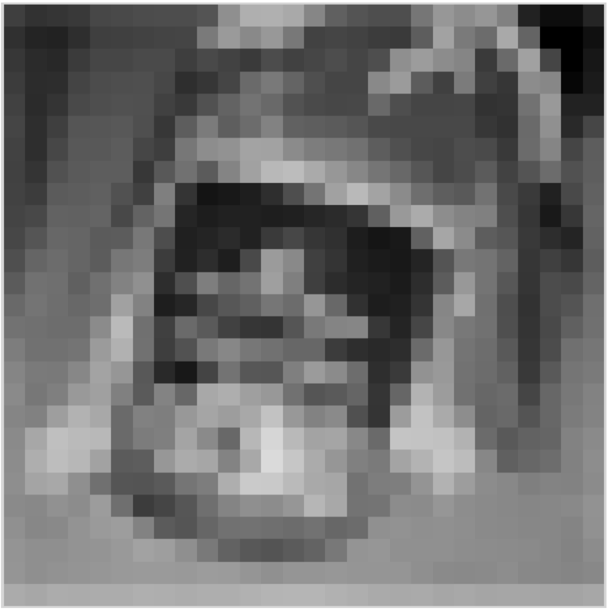
We can always go back to the array of numbers.

```
 48   47   60   71  121  105   64   89  125   97   18  108
 43   49   59   49   72   63   60   90   65   78   54  103
 51   65   59   85  123  118   92   61   50   65   77  128
 61   74   71   50   44   75  104  109   93   59   49  143
 79   82   88   38   57   71   21   40  106   64   45  145
 94  100  106   57   76   94   61   34  109   61   65  152
105  123   99   68   98   93   69   74  111   65   82  160
132  153  102  128  153  146   91  141  141   74   96  167
144  136   76  107  154  173  119  131  150  102  109  171
123  125  112   88   92   96  109  121  119  117  114  170
154  153  160  158  152  153  156  152  147  146  143  186
```

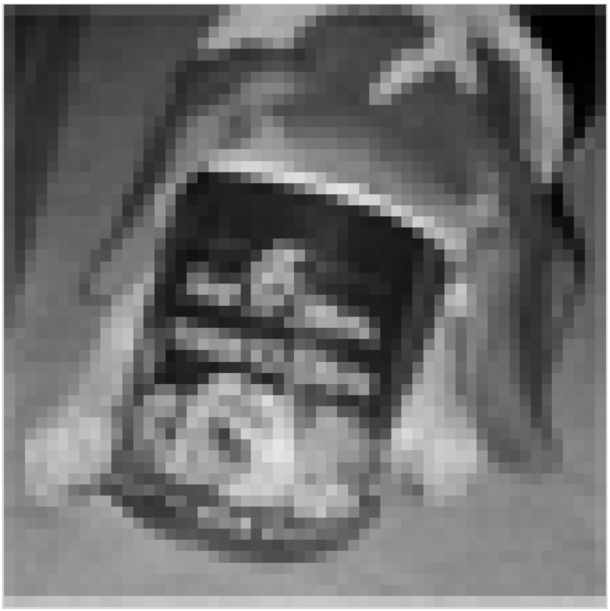
Here’s a larger array.
0=black, and 255=white.



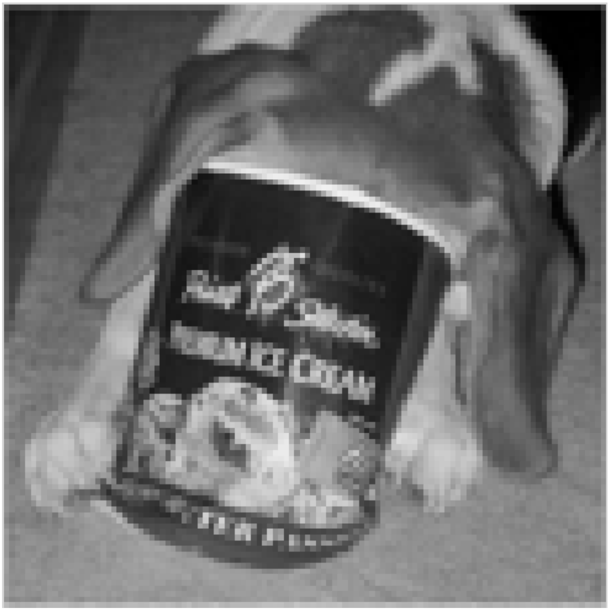
Here’s an even larger array. Now we have too many numbers to display on this screen.
0=black, and 255=white.



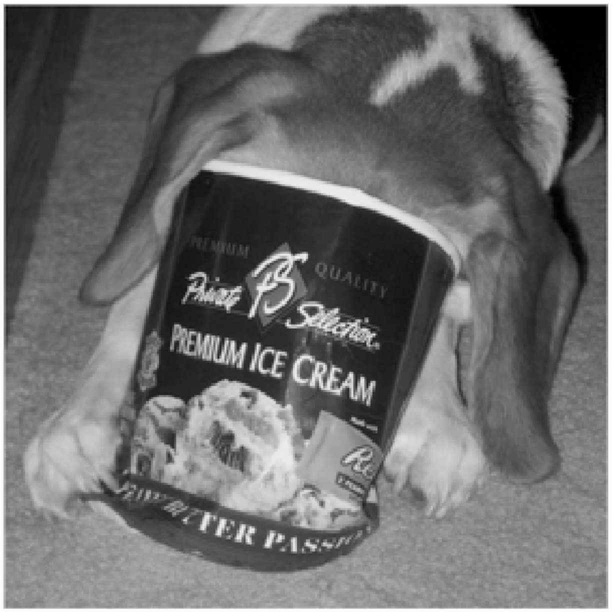
And larger …
0=black, and 255=white.



And larger
0=black, and 255=white.



Still larger. At this point, our eye can no longer discern most of the individual pixels.
0=black, and 255=white.



Largest.
0=black, and 255=white.

Images

Big Conclusion: Images are just 2D arrays that are displayed in an interesting way!

At some point, your eye can no longer distinguish the individual pixels.

In Python, images and matrices are exactly the same.

Color Image

**A color image is three different arrays.
The computer displays one of the arrays for red, one for green, and one for blue.**

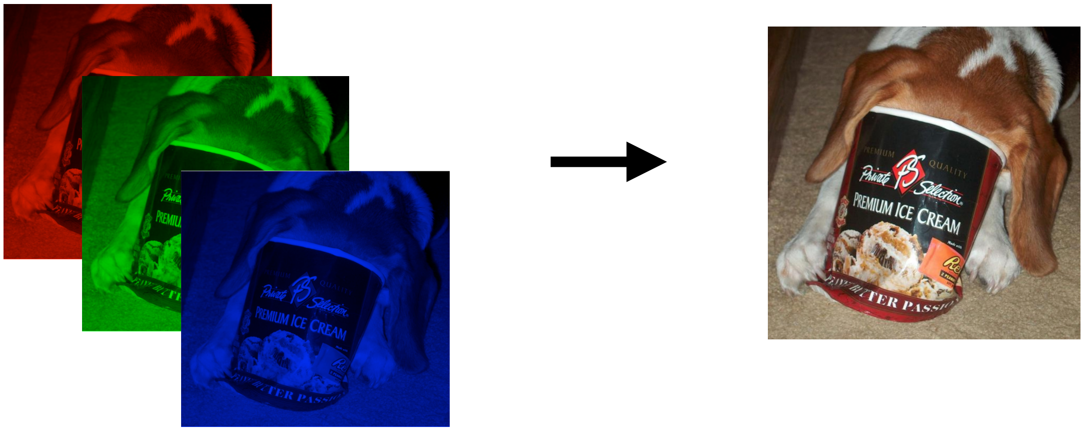


Image Files

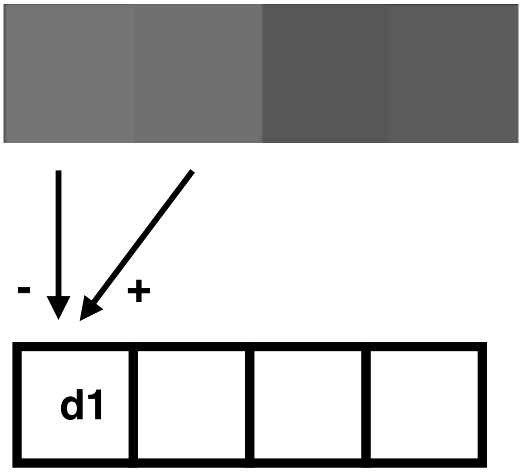
***.jpg, *.png, *.gif, *.bmp are all types of image files**

We may discuss the differences between these file types later in the class.

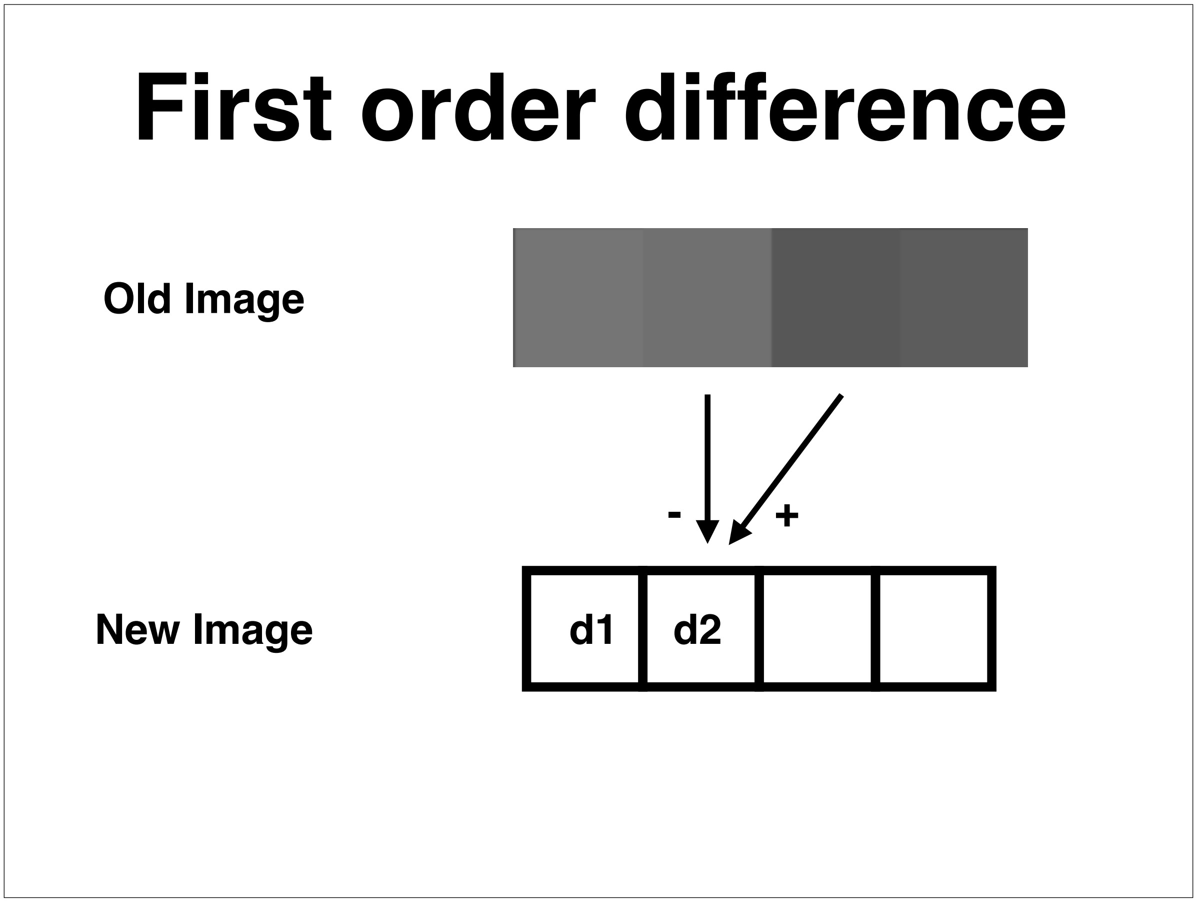
We’ll discuss some simple image processing algorithms.

First order difference

Old Image



New Image



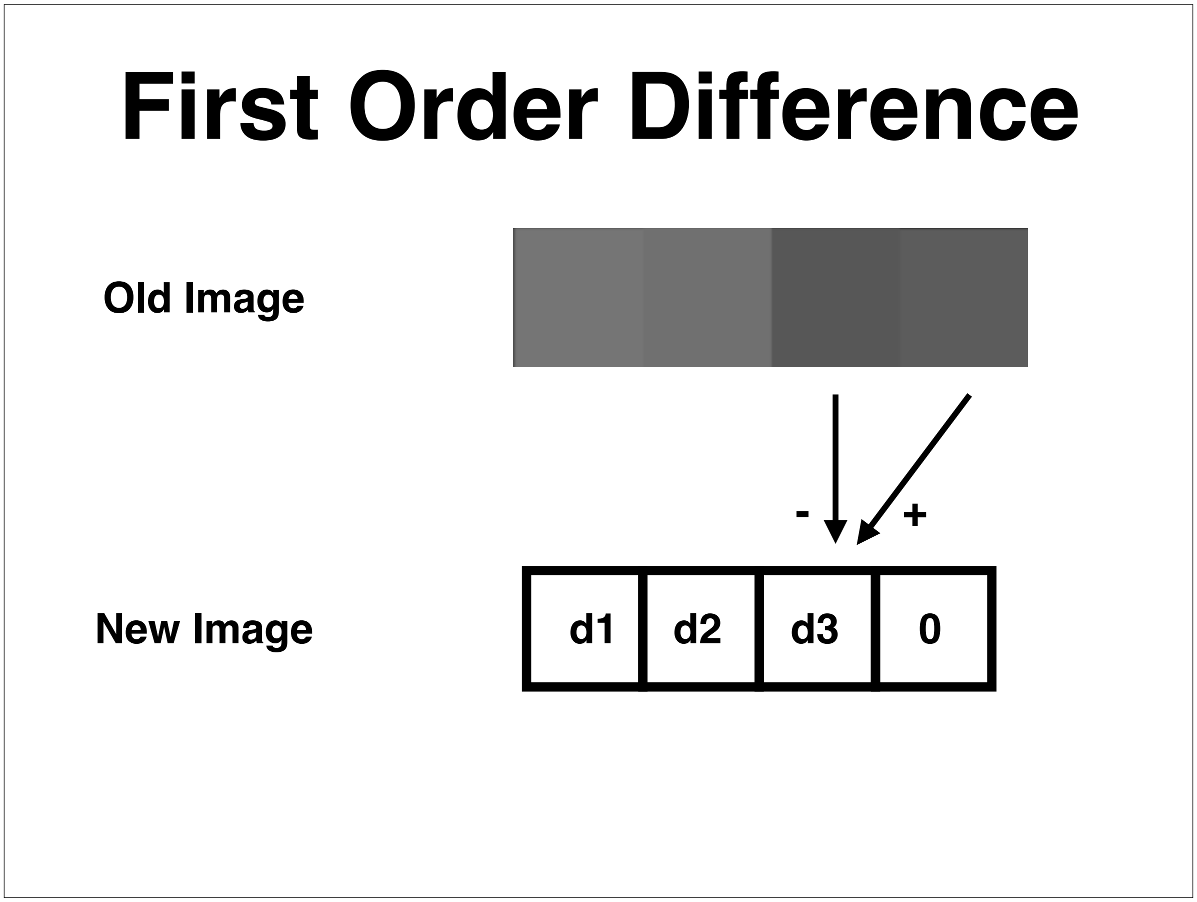
First order difference

Old Image

- +

New Image

d1	d2		



First Order Difference

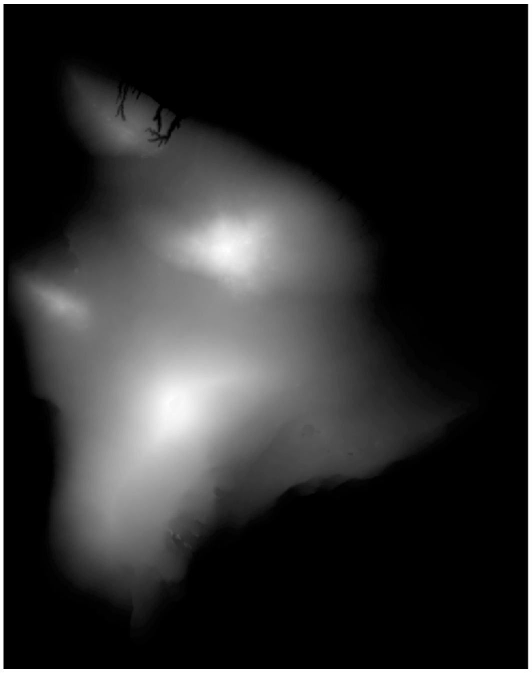
Old Image

- +

New Image

d1	d2	d3	0

Digital Elevation Map



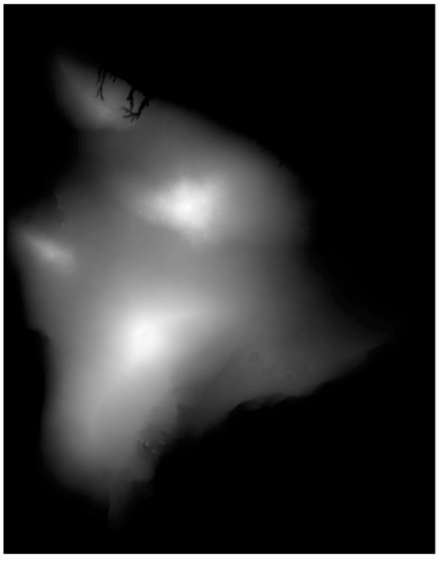
Each pixel is a number designating the location’s height.

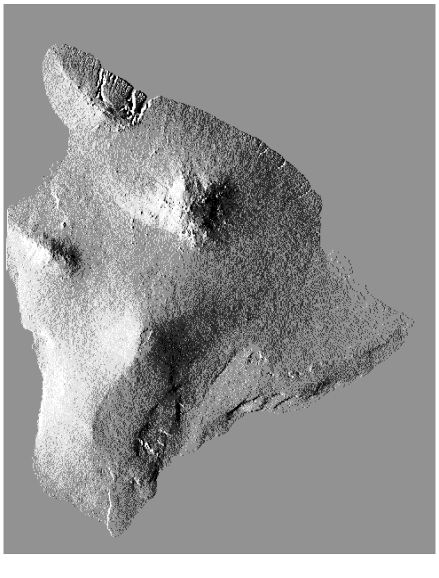
The brighter the pixel, the higher the point.

Hawaii

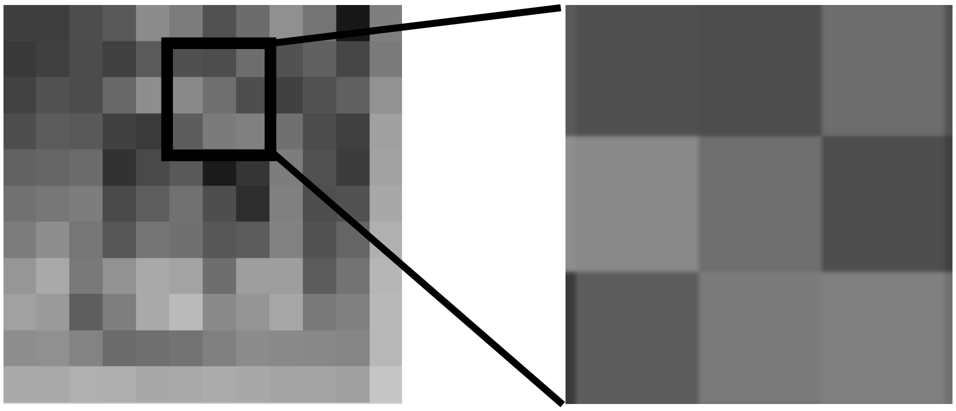
Relief Distortion Map

First order difference applied to Digital Elevation Map



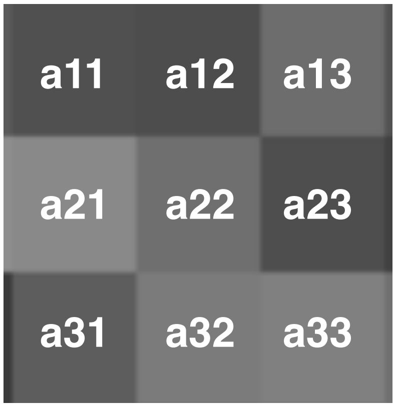


Isolating a small region



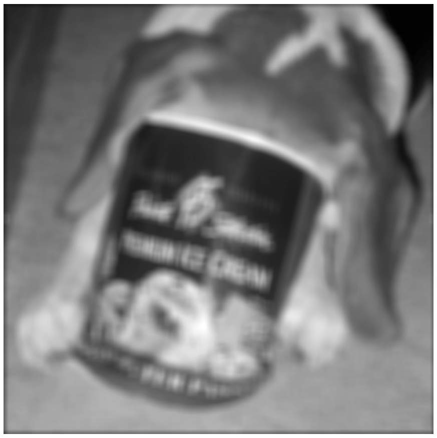
Mean of Square Region

Mean = $$\frac{1}{9}\left(a_{11} + a_{12} + a_{13} + a_{21} + a_{22} + a_{23} + a_{31} + a_{32} + a_{33}\right)$$



Mean Filtering

Replace each pixel with its local mean.



25x25 pixel kernel

Also called “Box Car Averaging”

Mean

Image with values *v* :

v_1	v_2							v_N

Weights:

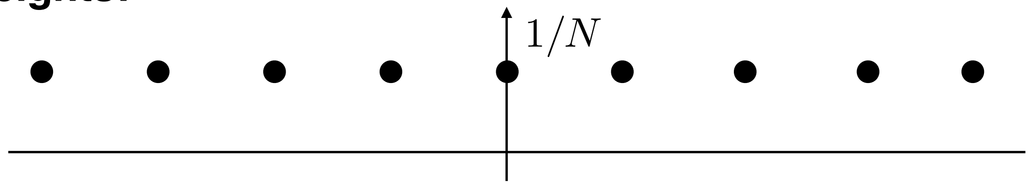
$$1/N \quad 1/N \quad 1/N \qquad \cdots \qquad 1/N$$

Mean

Image with values *v* **:**

v_1	v_2							v_N

Weights:



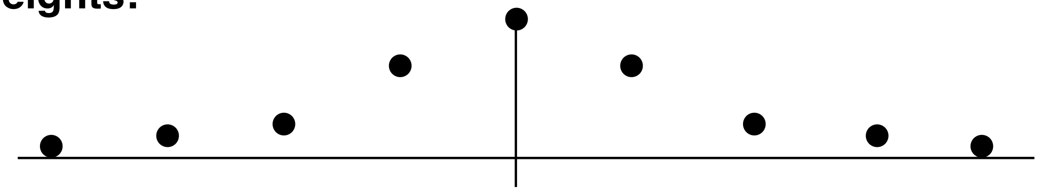
This seems weird. The values in the middle should matter more than values far away.

Weighted Mean

Rather than weighting each point equally, weight them differently.

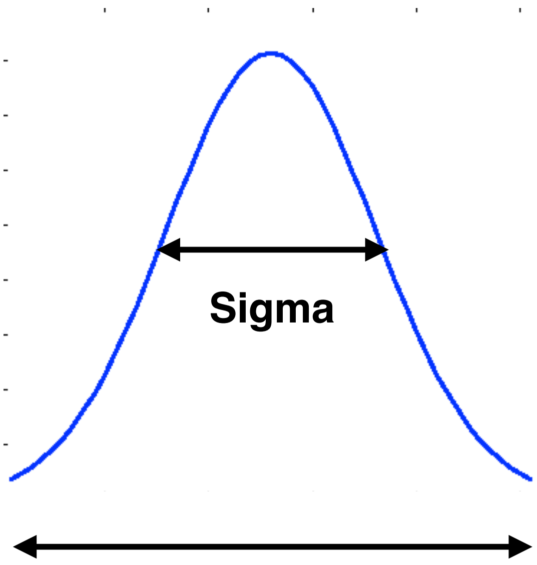
v_1	v_2							v_N

Weights:



Modifying the weights can solve this.
A Gaussian function is a good choice (fspecial **in Matlab).**

Gaussian Function



Sigma

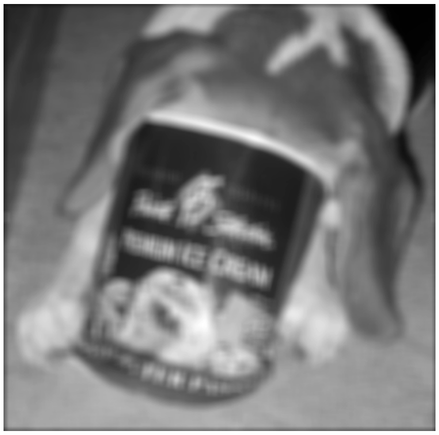
Size of Kernel

Sigma tells you how flat the weights are.
The higher the sigma, the flatter the weights.

The size of the kernel tells you how many pixels you're including.

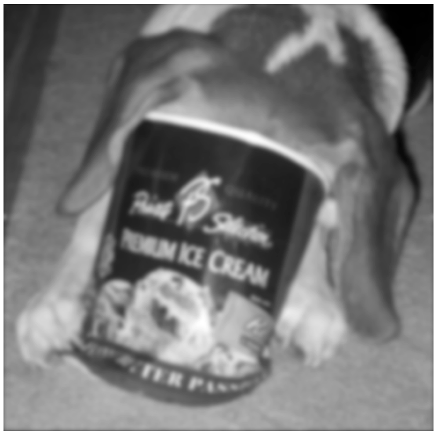
Weighted Mean Filtering

Box Car Filter



25x25 pixel kernel

Gaussian Filter

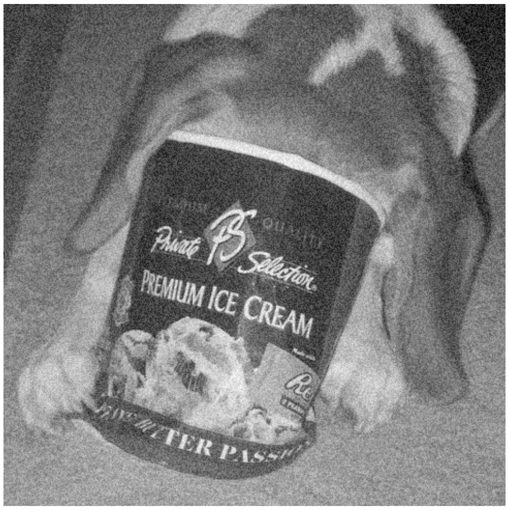


25x25 pixel kernel
sigma = 5

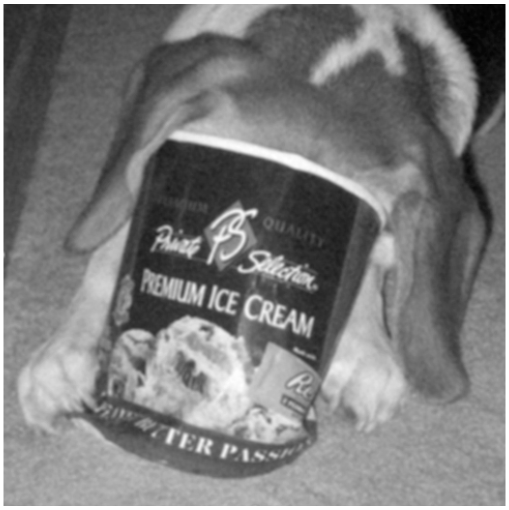
Gaussian Filtering retains a lot more of the information.

Image Denoising

Noisy Image



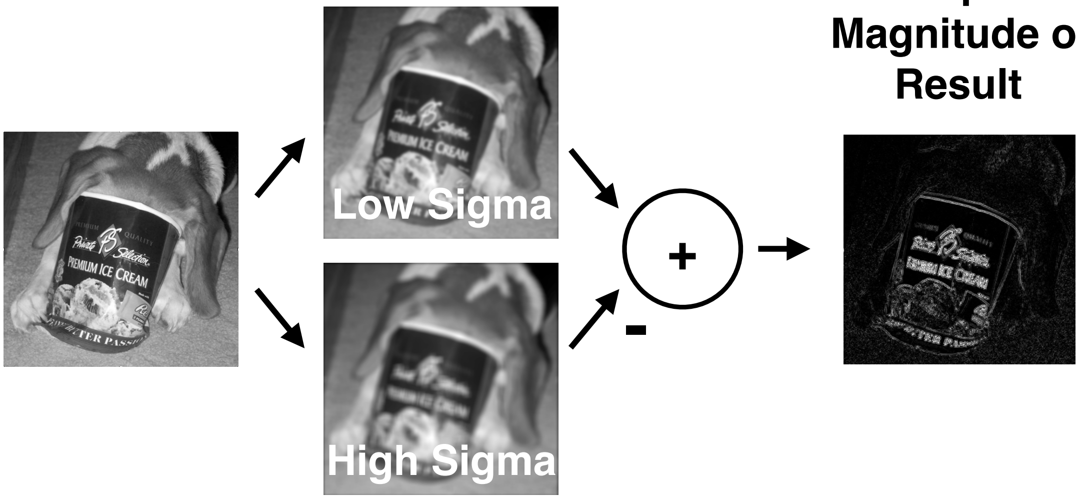
Gaussian Filter



9x9 pixel kernel
sigma = 3

Difference of Gaussians

Finds interesting points in the image.



Compute two Gaussian Filtered Images

Compute Magnitude of Result

Interesting pixels are bright.