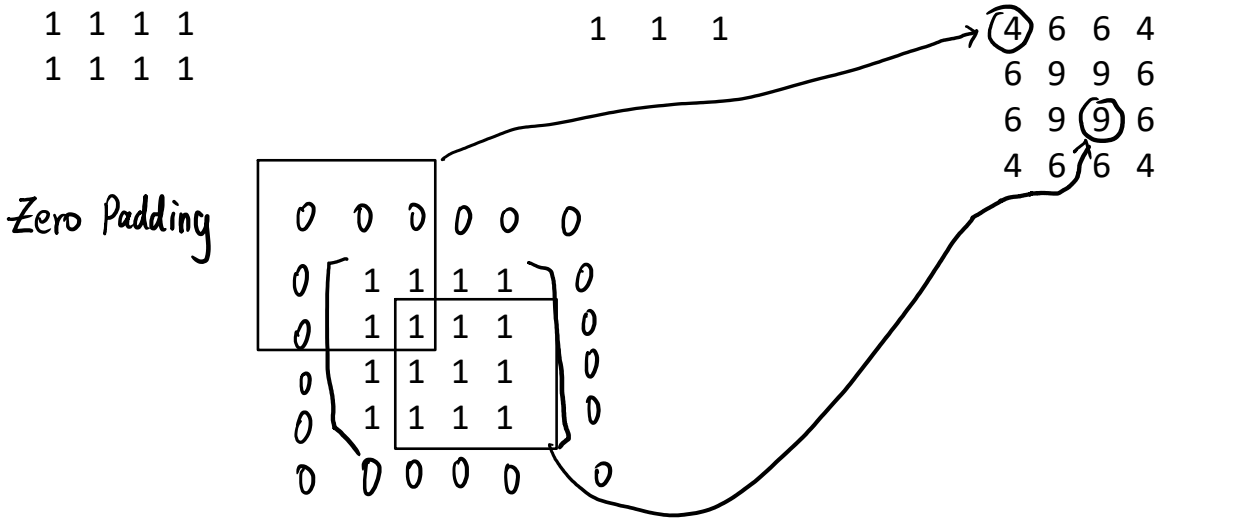
Lecture 14

Linear Filtering

```
>> I = uint8(ones(4,4));
I =
  4×4 uint8 matrix
   1   1   1   1
   1   1   1   1
   1   1   1   1
   1   1   1   1
```

```
>> h = ones(3,3)

h =
     1     1     1
     1     1     1
     1     1     1
```

```
>> J = imfilter(I, h)

J =
  4×4 uint8 matrix
    4   6   6   4
    6   9   9   6
    6   9   9   6
    4   6   6   4
```




```
>> I2 = 2*I;

I2 =
  4×4 uint8 matrix

   2   2   2   2
   2   2   2   2
   2   2   2   2
   2   2   2   2
```

```
>> J2 = imfilter(I2, h)

J2 =
  4×4 uint8 matrix

    8   12   12    8
   12   18   18   12
   12   18   18   12
    8   12   12    8
```

- Non-linear Filtering

Example:

```
>> X = [10 15 20 20 20 20 20 25 100]
X =
   10    15    20    20    20    20    20    25   100
>> mean(X)
ans =
   27.7778
```

```
>> median(X)

ans =
    20
```

```
>> Y = [10 15 20 20 20 20 20 25 500]
Y =
    10    15    20    20    20    20    20    25   500
>> mean(Y)
ans =
   72.2222
```

```
>> median(Y)
ans =
    20
```

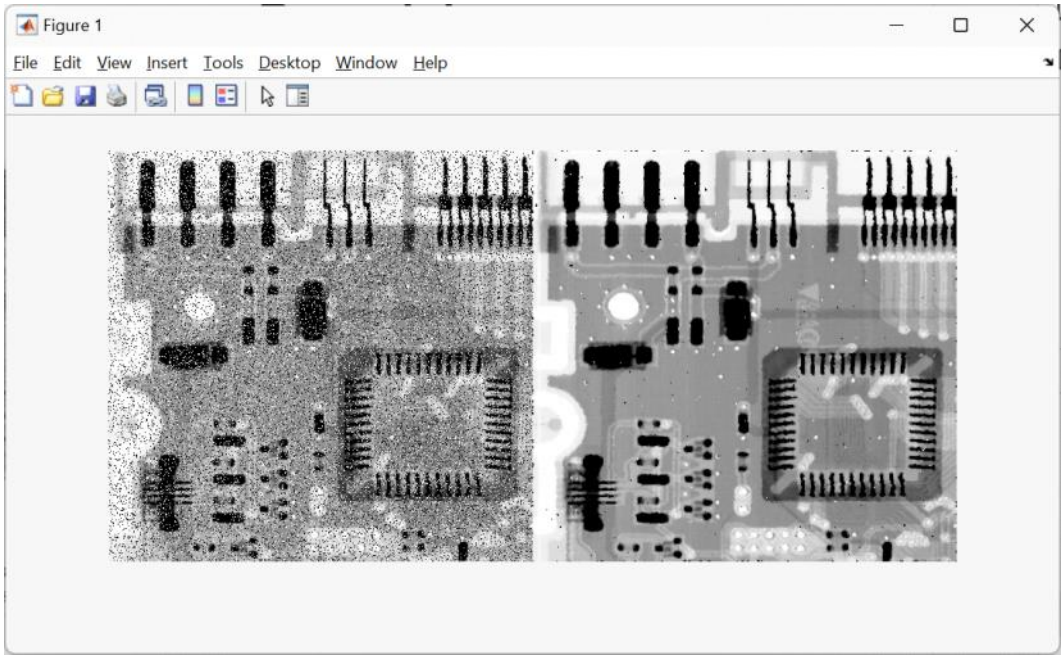
- The median ζ of a set of values is such that half the values in the set are less than or equal to ζ and half are greater than or equal to ζ
- In order to perform median filtering at a point in an image, we first sort the values of the pixel in the neighborhood, determine the median, and assign the value to the corresponding pixel in the filtered image.
- For example, in a 3×3 neighborhood, the median is the 5^{th} largest value.
- (10, 15, 20, 20, 20, 20, 20, 25, 100) results in a median of 20.
- The median filters force points with distinct intensity levels to be more like their neighbors.

medfilt2

2-D median filtering

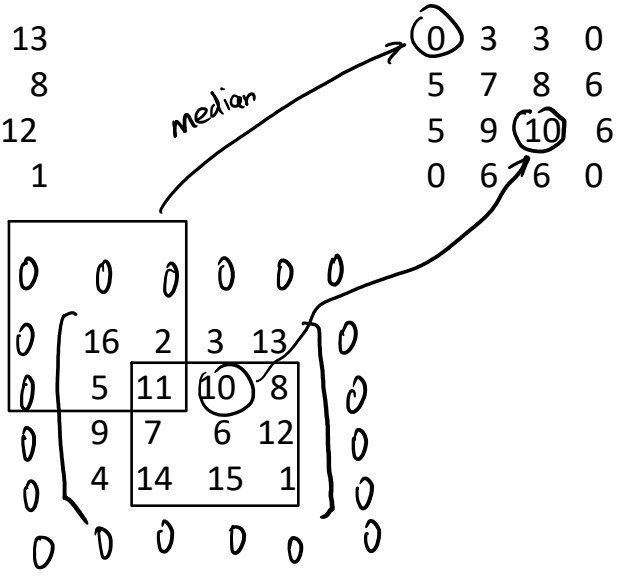
J = medfilt2(I) performs median filtering of the image I in two dimensions. Each output pixel contains the median value in a 3-by-3 neighborhood around the corresponding pixel in the input image.

```
>> I = imread('Fig0335(a)(ckt_board_saltpep_prob_pt05).tif');
>> K = medfilt2(I);
>> doc medfilt2
>> figure; imshowpair(I, K, 'montage')
```



```
>> I = uint8(magic(4))

I =

  4×4 uint8 matrix

   16   2   3  13
    5  11  10   8
    9   7   6  12
    4  14  15   1
```

```
>> K = medfilt2(I)

K =

  4×4 uint8 matrix

    0   3   3   0
    5   7   8   6
    5   9  10   6
    0   6   6   0
```



- Midterm Exam
  October 15, 2024, (Tuesday), in-class, 1:00 pm - 2:20 pm
  closed-book and closed notes.
  Allowed: one formula sheet (two sided), and a calculator.

  Lecture 1 -- Lecture 14,
  HW1, HW2.

  Example Topics:

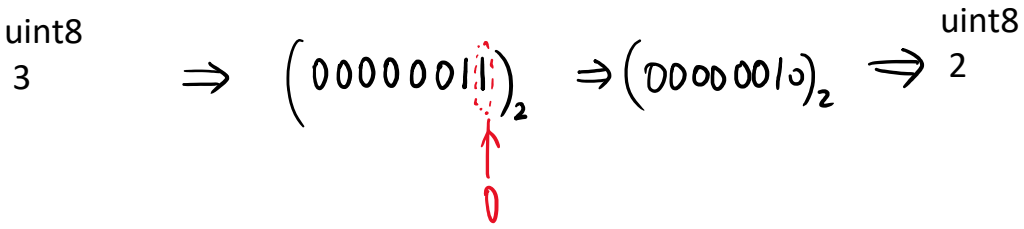
  bitplane processing:

  Given a pixel value in decimal, convert the value to binary representation, extract bit values.

```
>> A = uint8(3)v
A =
  uint8
   3
```

$\Rightarrow (00000011) \Rightarrow (00000010)_2 \Rightarrow$ 2

```
>> B = bitset(A,1,0)
B =
  uint8
   2
```


uint8
3
(00000011)2
(00000010)2
uint8
2

Geometric Spatial Transformation

Translation:

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 20 & 20 & 1 \end{bmatrix}$$

$$[x \quad y \quad 1] = [v \quad w \quad 1]\, T = [v \quad w \quad 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 20 & 20 & 1 \end{bmatrix}$$

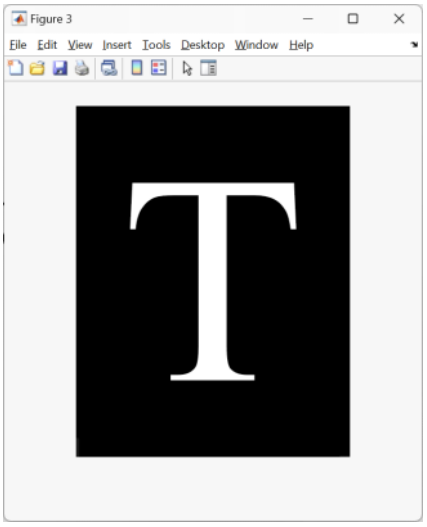
$$= [v+20 \quad w+20 \quad 1]$$

Scaling:

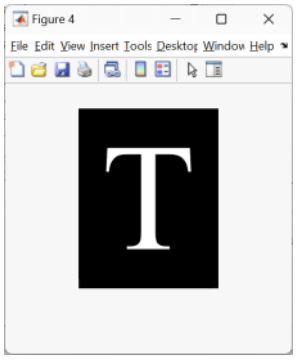
$$\begin{cases} x = 0.5v \\ y = 0.5w \end{cases} \qquad T = ? \qquad T = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
>>  I = imread('Fig0236(a)(letter_T).tif');
>> T = [0.5 0 0;0 0.5 0;0 0 1]

T =

   0.5000         0         0
        0    0.5000         0
        0         0    1.0000

>>  tform = affinetform2d(T);
>>  J = imwarp(I, tform);
```

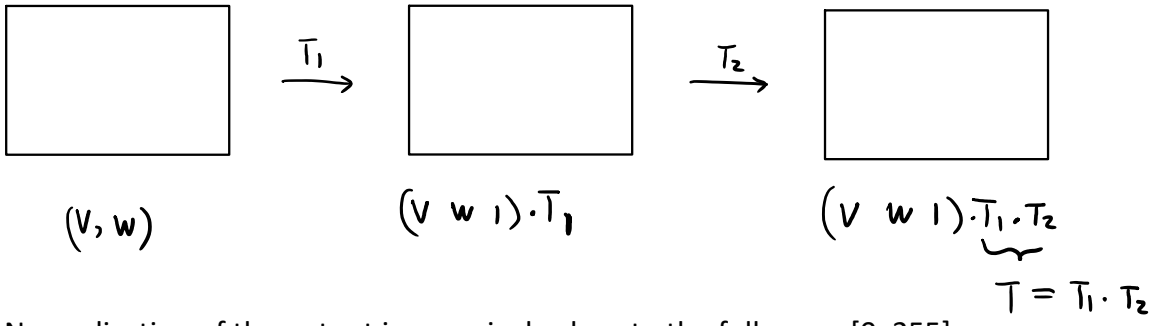
I



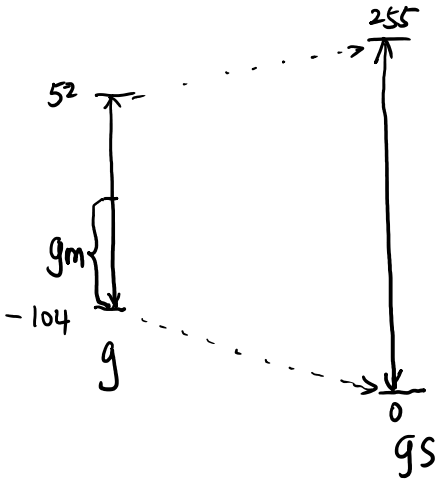
J



Sequence of transformations



- Normalization of the output image pixel values to the full range [0, 255]



- Output images should be normalized to the range of $[0, 255]$.

$$f_m = f - \min(f)$$
$$f_s = K[f_m / \max(f_m)]$$

- Histogram Equalization
- Spatial Filtering
- Boundary Padding Options