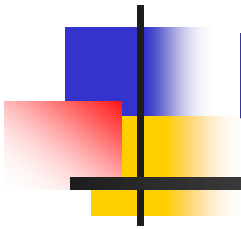
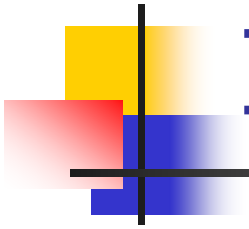


IMAGE PROCESSING USING MATLAB

Scott Bezan

bezansa@mcmaster.ca

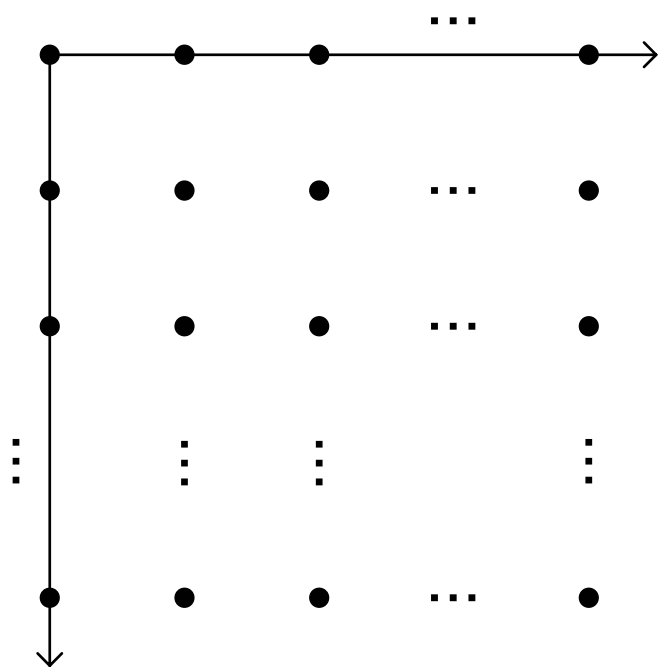
ITB 134

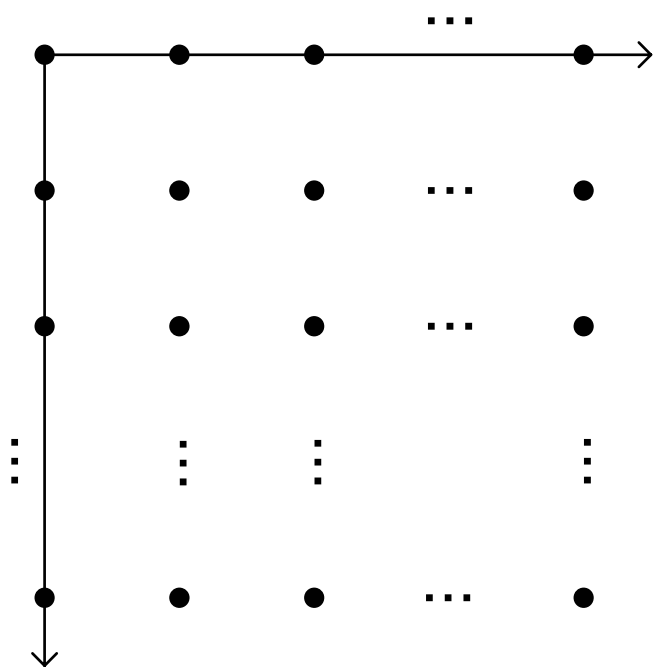


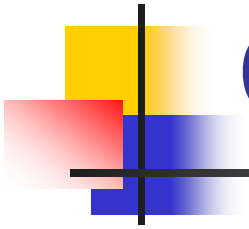
Introduction

- MAT – matrix
- LAB – lab(oratory)
- Images are just matrices (arrays) where the elements mean something (visually).
- Each pixel in an image has an intensity value.
- Arrays of size MxN.
- Each element is a pixel.

Coordinate conventions







Coordinates continued...

- Note indices in MATLAB begin with 1 and can only be positive real integers.
- (x,y) -> (r,c)
- (0,0) -> (1,1)
- Spatial coordinates have x as cols. and y as rows.

Images as matrices

Matrix

$$a = \begin{bmatrix} a(1,1) & a(1,2) & \cdots & a(1,N) \\ a(2,1) & a(2,2) & \cdots & a(2,N) \\ \vdots & \vdots & \ddots & \vdots \\ a(M,1) & a(M,2) & \cdots & a(M,N) \end{bmatrix}$$

Row vector

$$r = \begin{bmatrix} r_1 & \cdots & r_N \end{bmatrix}$$

Column vector

$$c = \begin{bmatrix} c_1 \\ \vdots \\ c_M \end{bmatrix}$$

Basic MATLAB functionality

- Creating a row vector

```
a=[1 2 3]
```

```
a=1 2 3
>>
```

- Creating a column vector

```
b=[1;2;3]
```

```
b=1
  2
  3
>>
```

- Creating a matrix

```
C=[1 2 3;4 5 6;7 8 9]
```

```
C=1 2 3
  4 5 6
  7 8 9
>>
```

Vector and matrix indexing

- To index an element from a vector

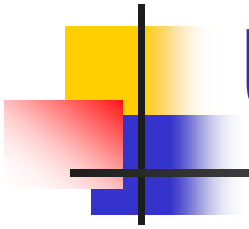
```
a(2) >> 2          b(end) >> 3
```

- To index an element from a matrix

```
C(2,3) >> 6        C(1,end) >> 7
```

- The colon ":" operator will yield a specified range

```
C(3,:) >> 7 8 9      2   C(:,2) >>
                     5
                     8
```



Useful functions

`length(X)`

- Returns the length of the matrix X.

`[M,N]=size(X)`

- Returns the length of the rows and columns (respectively) of X in separate variables.

`sum(X)`

- sums the elements in X (col. wise for matrices. Can specify the dimension.)

`X(:)`

- Selects elements of an array (col. by col. basis) and stacks them one atop the other.

```
sum(X(:)) <==> sum(sum(X))
```

`zeros(MxN)`

- Returns an MxN array of zeros.

`ones(MxN)`

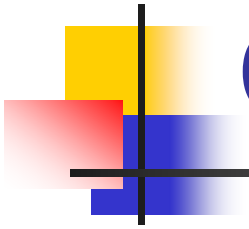
- Returns an MxN array of ones.

`rand(MxN)`

- Returns an MxN array of uniformly distributed random numbers from [0,1].

`randn(MxN)`

- Returns an MxN array of Gaussian distributed random numbers with mean 0 and unit variance.

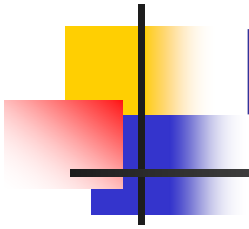


Operators

- Arithmetic
  - Perform numeric computations.
  - Matrix operations (linear algebra).
  - Array arithmetic operations (element by element) via dot (“.”) operator.

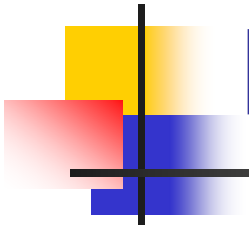
A and B are MxN matrices.

```
A*B   A.*B
```



More operators

- Relational
  - Compare operands quantitatively.
  - <, <=, > >=, ==, ~=
- Logical
  - AND, OR, NOT, XOR
  - Anything ~=0 is true otherwise is false.



Flow control

- If statements

```
if expression1
      statements1
elseif expression2
      statements2
else expression3
      statements3
end
```

- For loops

```
for index = start:increment**:end
     statements
end
(** no increment, assumed 1)
```

- While loops

```
while expression
     statements
end
```

- Switch

```
switch switch_expression
     case case_Expression
          statement(s)
     case{case_expression1,case_expression2,...}
          statement(s)
     otherwise
          statement(s)
end
```

```
break
```

- terminates loop in which it resides.

Image Processing Toolbox (IPT)

- Reading images
  - IPT accepts images of type TIFF, JPEG, GIF, BMP, PNG and XWD.
  - filename is string of complete filename (i.e. path (if not in pwd) and extension).
  - use quotes for strings!!

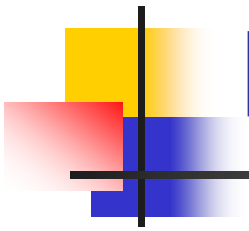
```
f=imread('picture.jpg');
f=imread('D:\images\picture2.jpg');
f=imread('.\images\picture3.jpg');
```

Reading images cont...

- Some types are not readable by imread (i.e. .bin).

```
fid=fopen('picture.bin');
f=fread(fid,[numRow, numCol]);
fclose(fid);
```

- Need to know the size of image beforehand.



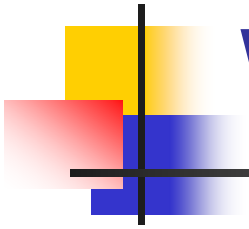
Displaying Images

`imshow(f,G)`

- f is the image array
- G is the number of intensity levels (default 256).

`colormap(gray(G));image(f)`

- careful of data type and max values**



Writing images

- For IPT compatible images,

```
imwrite(f,'filename')
imwrite(f,'filename','format')
```

- JPEG

```
imwrite(f,'picture.jpg','quality',q)
```

- q is quality factor 0<=q<=100
- lower q, higher distortion.

Exporting images from MATLAB

- In figure window
  - File -> Export

OR/

```
print -f # -d fileformat -r res# filename
```

- # is figure number
- fileformat MUST be allowable by IPT
- res# resolution in dpi

Data Classes

Class	Range	# bytes/element
double	$[-10^{308},10^{308}]$	8
uint8	[0,255]	1
uint16	[0,65535]	2
uint32	[0,4294967295]	4
int8	[-128,127]	1
int16	[-32768,32767]	2
int32	[-2147483648, 2147483647]	4
single	$[-10^{38},10^{38}]$	4
char	characters	2
logical	[0,1]	1

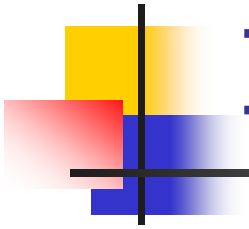
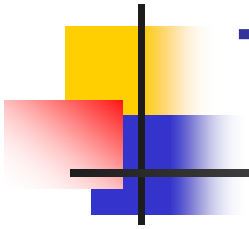


Image types

- Intensity
  - data matrix with entries scaled to intensities.
  - uint8 -> [0,255]
  - uint16 -> [0,65535]
  - double -> [0,1] (*scaled)
- Binary
  - logical array of 1’s and 0’s.



Type casting

- convert from one data type to another.

  ```
  B=data_class_name(A)
  ```

  converts to array A to type of data_class_name.

- MATLAB expects operands in numeric computations to be of double precision floating point numbers.

Image classes ⇔ Image types

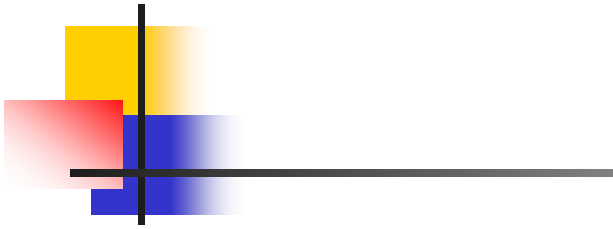
- Perform necessary scaling to convert between image classes and types.

`im2uint8(X)`

- detects input data class and scales to allow recognition of data as valid image data.

`mat2gray(X,[Amin,Amax])`

- takes arbitrary double array input and scales to range [0,1].
- values < Amin =>0, >Amax =>1



`im2double(X)`

- converts X input to class double in range [0,1] unless input is of class double, no effect.

`im2bw(X,T)`

- converts intensity image to a binary image.
- anything less than T output set to 0, otherwise output set to 1.
- T=[0,1]
- output is logical

Code optimization

- Can eliminate loops by using vectors as arguments.

```
for x=1:50000
        f(x)=A*sin((x-1))/(2*pi));
end
```

t=12.6870 s

⇔

```
x=0:49999
f=A*sin(x/(2*pi));
```

t=0.0160 s

- Method 2 is almost 800 times faster!!!