Ground Zero

(COMPONENTS OF MATLAB)

① → Workspace browser
→ command history
→ command window
→ figure
→ m-file

→ HELP images
→ HELP

② Set Path, Save, Clear, SINE WAVE EXAMPLE -splot

③ Coordinate System for digital images

2 stages of digitization → sampling, digitizes coordinate values

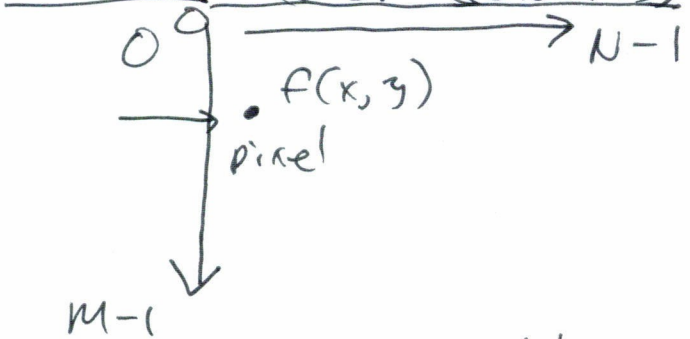
$f_s > 2B$

$B < \frac{f_s}{2}$

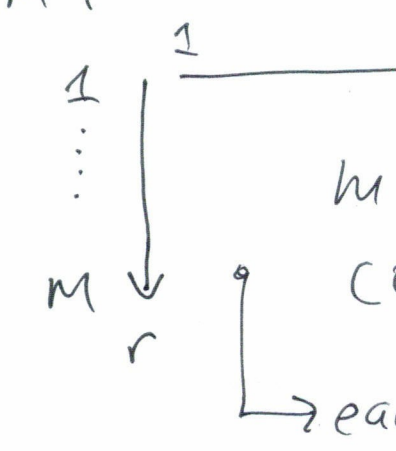
quantization, amplitude values

Image | size (M rows) × (N columns)

$f(x, y)$ ← spatial domain



$$= \begin{bmatrix} f(0,0) & \cdots & f(0,N-1) \\ \vdots & & \\ f(M-1,0) & \cdots & f(M-1, N-1) \end{bmatrix}$$



MATLAB CONVENTION

→ each element corresponds to one pixel

Types of Images

- binary → '0', '1' only for pixel values (uint8) → logical flag / logical
- intensity → double [0, 1] or uint8 [0, 255], single matrix
- color (RGB) → M x N x 3 array
- indexed → M x N array, M x 3 vector (colormap)

Useful Image Functions

- Reading → imread()
- Writing → imwrite()
- Info → imfinfo()
- Convert to grayscale from color → rgb2gray() ← HERE
- Convert matrix to gray (w/ scaling) → mat2gray()
- im2uint8 → double to uint8
- im2bw
- DISPLAY → imshow( )

find connected

1*

4-connected example

- SHOW EXAMPLES

MOVE TO END

K = imfinfo(.jpg)
K = iminfo(.bmp)

JPG quality example

2*

image bytes = K.width * K.height * K.BitDepth * 8

compr bytes = K.filesize

The Nitty Gritty

MATLAB naturally handles vectors and matrices...

VECTORS - `v = [1 3 5 7 9]`, `V = 1:2:9;`

- `w = v'`

- `v • w` (inner) ⇒ dot product
  1×5 5×1

- `w . v` → outer product
  5×1 1×5 5×5

Indexing

- `v(2)`, `v(2:4)`, `v(3:end)`, `v(end:-1:1)`

MATRICES `A = [1 2 3; 4 5 6; 7 8 9];` 3×3

~~A(2)~~

`B = [2 4 5; 3 1 2]` 2×3

- `BA` Matrix Multiply

- `A A` vs. `A .* A` (array processing)
  pointwise multiplication

  important when operating per-pixel

- **Indexing** `A(2,3)`

  `C3 = A(:,3)`

  `R2 = A(2,:)`

  submatrix

  `T2 = A(1:2, 2:3)`

- Stacking into a column vector

```
V = A(:)
```

- SUMMING ELEMENTS OF A MATRIX

```
sum(A(:))   or   sum(sum(A))
```

- Standard Matrices

```
zeros()
ones()
rand()
randn()
magic()
```

- Simple function

```
function [a, b] = imrange(f)
% returns min and max element
% values of matrix f

a = min(f(:));
b = max(f(:));
```

- The USUAL COMPARATORS

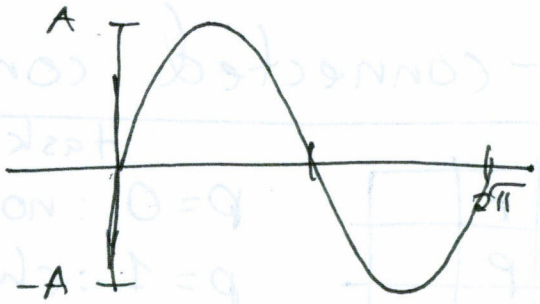
```
==   equal to          &, |, ~
~=   isnot equal       AND, OR, NOT
```

- the USUAL CONTROL STATEMENTS

IF, FOR, WHILE ⟶ END

- Vectorizing Loops (1-D)

$$f(x) = A \sin\left(\frac{x}{2\pi}\right)$$



WRONG WAY

efficiency

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

RIGHT WAY

```
x = 1:m
f = A * sin(x/(2*pi));
```

- IN 2-D,

```
[C, R] = meshgrid(c, r)
```

length(c) x length(r) matrices

row vectors

Show 2-D sin example from book (OPEN CODE)

```
[rt, f, g] = twodsin(1, 1/(4*pi), 1/(4*pi), 512, 512);
imabsdiff(f,g)
```

Cells & structs

```
c = {'hi', 3, A}    c{3}
d.name = 'title';
```