Notation and Preliminaries

Notation
Useful definitions
What is an image?
Image representations: screen, matrix, Matlab
Spatial frequency units

Notation: vectors and matrices

Vector in R^N (N-dimensional vector space):

$$\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_N \end{bmatrix}^T$$

$M \times N$ matrix ($R^N \to R^M$):

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & x_{1N} \\ a_{21} & a_{22} & \dots & x_{2N} \\ \vdots & \vdots & & \vdots \\ a_{M1} & a_{M2} & \dots & x_{MN} \end{bmatrix}$$

Matrix multiplication: $\boldsymbol{C} = \boldsymbol{AB}$, $c_{ij} = \sum_{k=1}^{N} a_{ik} b_{kj}$

Matrix product is not, in general, commutative: $\boldsymbol{AB} \neq \boldsymbol{BA}$

Transpose of matrix product: $(\boldsymbol{AB})^T = \boldsymbol{B}^T \boldsymbol{A}^T$

Inverse of matrix product (square matrices only): $(\boldsymbol{AB})^{-1} = \boldsymbol{B}^{-1} \boldsymbol{A}^{-1}$

We will use some *matrix calculus*, so review multivariable calculus material (e.g., MA225)

Some special matrices

1. Symmetric matrix: $\boldsymbol{A}$ is real and $\boldsymbol{A} = \boldsymbol{A}^T$
2. Hermitian matrix: $\boldsymbol{A}$ is complex and $\boldsymbol{A} = (\boldsymbol{A}^*)^T \stackrel{\triangle}{=} \boldsymbol{A}^H$:

$$[\boldsymbol{A}]_{ij} = a_{ij} \Rightarrow [\boldsymbol{A}^H]_{ij} = a_{ji}^*$$

   $\boldsymbol{A}^*$ is a conjugate of matrix $\boldsymbol{A}$ (element by element).
   $\boldsymbol{A}^H$ is also called a conjugate-transpose of $\boldsymbol{A}$.

3. Orthogonal matrix: $\boldsymbol{A}$ is real and $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix (1's on the main diagonal and 0's otherwise)
4. Unitary matrix: $\boldsymbol{A}(\boldsymbol{A}^*)^T = (\boldsymbol{A}^*)^T\boldsymbol{A} = \boldsymbol{I}$

Any real, orthogonal matrix is unitary but not vice-versa.

5. **Toeplitz matrix** (constant value on each diagonal):

$$\begin{bmatrix} t_0 & t_{-1} & t_{-2} & \dots & t_{-N+1} \\ t_1 & t_0 & t_{-1} & \dots & t_{-N+2} \\ t_2 & t_1 & t_0 & \dots & t_{-N+3} \\ \vdots & \vdots & \vdots & & \vdots \\ t_{N-1} & t_{N-2} & t_{N-3} & \dots & t_0 \end{bmatrix}$$

Example: if $t_k = t_{-k}$, Toeplitz matrix describes autocorrelation of a *stationary* stochastic process

6. **Circulant matrix** (circular shift between rows – special case of Toeplitz matrix):

$$\begin{bmatrix} c_0 & c_1 & c_2 & \dots & c_{N-1} \\ c_{N-1} & c_0 & c_1 & \dots & c_{N-2} \\ c_{N-2} & c_{N-1} & c_0 & \dots & c_{N-3} \\ \vdots & \vdots & \vdots & & \vdots \\ c_1 & c_2 & c_3 & \dots & c_0 \end{bmatrix}$$

Example: implementation of *circular convolution* of discrete-time signals

Other useful definitions

For vectors $\boldsymbol{x}, \boldsymbol{y} \in R^N$:

1. Inner (scalar) product:
$$\boldsymbol{x} \cdot \boldsymbol{y} = \boldsymbol{x}^T \boldsymbol{y} = \boldsymbol{y}^T \boldsymbol{x} = \sum_{i=1}^{N} x_i y_i.$$

2. Outer product:
$$\boldsymbol{x}\boldsymbol{y}^T = \begin{bmatrix} x_1 y_1 & \dots & x_1 y_N \\ \vdots & & \vdots \\ x_N y_1 & \dots & x_N y_N \end{bmatrix}$$

3. Vector norm (general form):
$$\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{N} |x_i|^p \right)^{1/p}$$

Three examples of vector norm:

- City-block/Taxi-cab/Manhattan norm ($p = 1$), also called L1 norm:

$$\|\boldsymbol{x}\|_1 = \sum_{i=1}^{N} |x_i|$$

- Euclidean norm ($p = 2$), also called L2 norm:

$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_{i=1}^{N} |x_i|^2} = \sqrt{\boldsymbol{x} \cdot \boldsymbol{x}} = \sqrt{\boldsymbol{x}^T \boldsymbol{x}}$$

- Maximum norm, also known as Chebyshev norm ($p = \infty$):

$$\|\boldsymbol{x}\|_\infty = \max(|x_1|, |x_2|, ..., |x_N|)$$

4. Orthogonality of vectors:

$$\boldsymbol{x} \perp \boldsymbol{y} \text{ iff } \boldsymbol{x} \cdot \boldsymbol{y} = 0.$$

5. Linear independence:
   - vectors: $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_k \in R^N$
   - scalars: $c_1, c_2, ..., c_k \in R$

   If $c_1\boldsymbol{x}_1 + c_2\boldsymbol{x}_2 + ... + c_k\boldsymbol{x}_k = \boldsymbol{0}$, where $\boldsymbol{0}$ is the zero vector, only if $c_1 = c_2 = ... = c_k = 0$, then vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_k$ are called *linearly independent*.

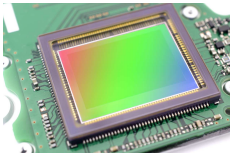
What is an image ?

An image is defined as light:

- emitted by objects, and/or
- emitted by source and then reflected or transmitted by objects,

into the field of view of a sensor:

- *film* – light triggers photochemical reaction,
- *vacuum tube* (vidicon, plumbicon) – light triggers electron departure from photocathode, thus inducing current,
- *CCD* (charge-coupled device) – light induces electrical charge,
- *CMOS* (complementary metal-oxide-semiconductor) – light induces voltage.



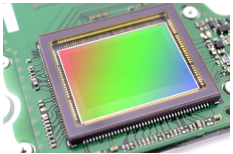
What is an image ?

An image is defined as light:

- emitted by objects, and/or
- emitted by source and then reflected or transmitted by objects,

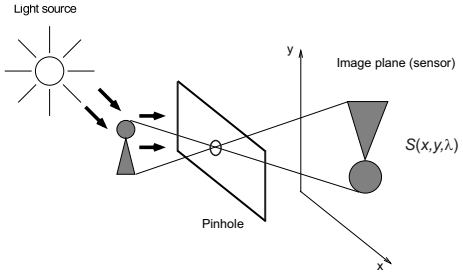
into the field of view of a sensor:

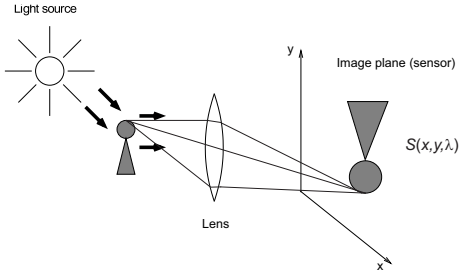
- *film* – light triggers photochemical reaction,
- *vacuum tube* (vidicon, plumbicon) – light triggers electron departure from photocathode, thus inducing current,
- *CCD* (charge-coupled device) – light induces electrical charge,
- *CMOS* (complementary metal-oxide-semiconductor) – light induces voltage.



What is the size of a typical sensing area in each case?

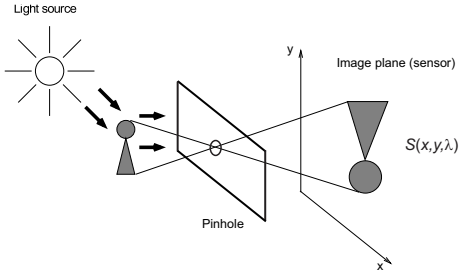
How is an image captured?

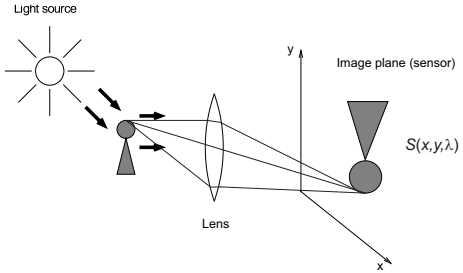




Why is a lens needed?

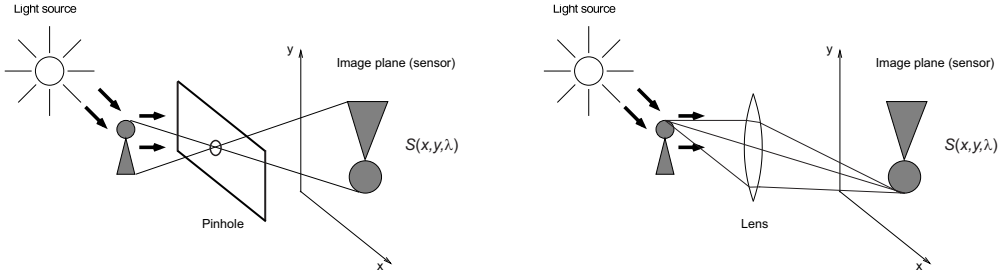
How is an image captured?





Why is a lens needed? Homework #1 will help answer this question.

How is an image captured?



Why is a lens needed? Homework #1 will help answer this question.

Light captured by the sensor is a **2-D (spatial) function** that integrates all wavelengths:

$$I(x,y) = \int_0^\infty S(x,y,\lambda)V(\lambda)d\lambda, \tag{1}$$

where $\boldsymbol{x} = [x,y]^T \in R^2$ denotes spatial location, $S(x,y,\lambda)$ is the spectral density of light projected onto sensor surface and $V(\lambda)$ is a sensitivity function of the sensor (how it reacts to various wavelengths λ or frequencies $1/\lambda$). $I(x,y)$ is often called **intensity** or **grayscale**.

More realistic images

Color images: Intensity (grayscale) only captures how bright/dark are parts of a scene. In order to represent color, 3 components are needed, for example red, green, and blue:

$$(I_R(x,y),\ I_G(x,y),\ I_B(x,y))$$

The use of three-component color representation is supported by empirical and physiological data (to be discussed later).

Multispectral imaging captures 7–10 narrow spectral (color) channels and is used in remote sensing as well as document and artwork analysis.

Hyperspectral imaging captures many more such channels (>200) and is used in space-based sensing.

In this course, we will focus on grayscale and three-component color images.

Image dimensions and aspect ratio

Due to the finite size of image sensor (e.g., CMOS), image $I(x, y)$ is usually observed in a rectangular window $\mathcal{W}$ with width W and height H.

The ratio W/H is called the *aspect ratio* and denoted $ar = W/H$.

The most common aspect ratio values are:

- $4/3 = 1.33$: standard definition TV, older computer monitors (640×480),
- 1.6 – 1.78: widescreen monitors (1,920×1,200, 2,560×1,440, ...),
- $16/9 = 1.78$: high definition TV (1,920×1,080),
- 2.2 and more: ultra-widescreen monitors, various cinema screens.

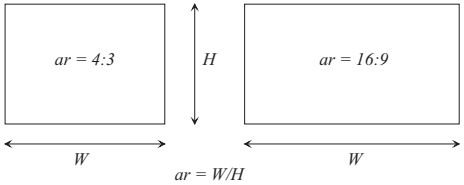


Image = 2-D function

An image is a distribution of light on sensor surface as shown in slide #8.

Mathematically, it is a continuous 2-D function $I_c(x_c, y_c)$ with $(x_c, y_c) \in R^2, I_c \in R^+$.

To **digitally** process this image, the intensity I_c must undergo:

1. *sampling*: continuous coordinates $(x_c, y_c) \rightarrow$ discrete coordinates $[m, n]$,
2. *quantization*: continuous amplitude $I_c \rightarrow$ discrete amplitude I.

$$I_c(x_c, y_c), \quad (x_c, y_c) \in R^2, I_c \in R^+$$

$$\Downarrow$$

$$I[m, n], \quad [m, n] \in Z^2, I \in Z^+$$

where Z = integers, Z^+ = non-negative integers. $I[m, n]$ is a **discretized image.**

Discretized image

$I(x_c, y_c) \rightarrow I[m, n]$, where m is the horizontal index and n is the vertical index. For example:

$$\begin{bmatrix} I[0,0] & I[1,0] & ... & I[W-1,0] \\ I[0,1] & I[1,1] & ... & I[W-1,1] \\ \vdots & \vdots & & \vdots \\ I[0,H-1] & I[1,H-1] & ... & I[W-1,H-1] \end{bmatrix}$$

This represents a $W \times H$ discrete image: W pixels per row and H rows per image.

A **pixel** may be understood either as a discrete position in the image, e.g., $m = 2, n = 1$, or a discrete position *and* the associated intensity $I[2, 2]$. The meaning is usually clear from the context but one has to be careful!

Although this looks like a matrix, the first index is running horizontally (along rows) !

Image representation: function versus matrix

Image as *2-D continuous function*: $I_c(x_c, y_c)$

- position of image point: x_c - horizontal, y_c - vertical

Image as *2-D discrete function*: $I[m, n]$, $m = 0, ..., W-1, n = 0, ..., H-1$

- position of pixel: m - horizontal, n - vertical
- $W \times H$ image is a 2-D discrete-space function (function notation)

Image as *2-D matrix*: $I[n, m]$, $n = 1, ..., H, m = 1, ..., W$

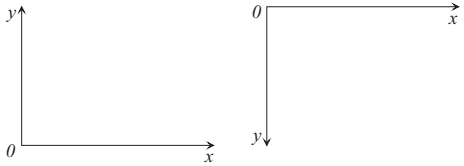
- $W \times H$ image becomes an $H \times W$ matrix in matrix notation
- the order of indexes is swapped between function and matrix notations
- this is very important for *Matlab* implementations
- another caveat: matrix indexes in *Matlab* start at 1, whereas in signal/image processing indexes usually start at 0 !!! One must remap in *Matlab* code.

Coordinate system

In both function and matrix representation of an image, the origin of the coordinate system is in the top-left corner.

This is called scanning-based coordinate system (with the y axis pointing down).

More traditional is the Cartesian coordinate system with the y axis pointing up.



Cartesian Scanning-based

We will primarily use the scanning-based system, that originates from the way images were scanned in analog TV (definition established in 1920s), since it is universally used in image processing. Also, *Matlab*'s `imshow` uses scanning-based convention. In some lectures, we will use the Cartesian system for convenience.

Final source of confusion

Matlab "understands" two dynamic ranges of image intensity:

- `uint8` $\Rightarrow [0, 255]$ – 0 is black and 255 is white,
- `float` $\Rightarrow [0.0, 1.0]$ – 0.0 is black and 1.0 is white.

One must be aware at all times which representation one uses.

For example, the function `imshow` to display images assumes that a `float` array contains luminance values from 0.0 to 1.0. If one stores values from 0.0 to 255.0 in this array, then values from 1.0 up will be interpreted as white and a typical image stored in this array will be displayed as a white rectangle. Conversely, if a `float` array with data in 0.0–1.0 range is copied to an *unit8* array, only 0 and 1 values will result and a black rectangle will be displayed.

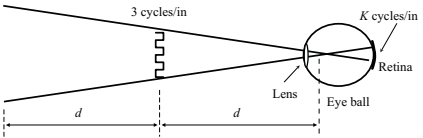
If you see a white or black rectangle as an output of your *Matlab* code on computer screen, suspect a data-range problem.

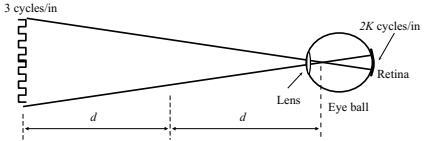
Spatial frequency units – problem

In time-dependent 1-D signals, the *sampling period* is expressed in units of time, e.g., in seconds [s], milli-seconds [ms], and the sampling frequency is expressed in cycles/second [Hz], thousands of cycles/second [kHz], ...

In 2-D signals, the spatial sampling period could be expressed in units of distance, e.g., in centimeters [cm] or inches [in], but then the spatial sampling frequency would have to be expressed in cycles/cm or cycles/in.

This would be cumbersome since a frequency of 3 cycles/in at distance d would be perceived as 6 cycles/in at distance $2d$ due to perspective projection.



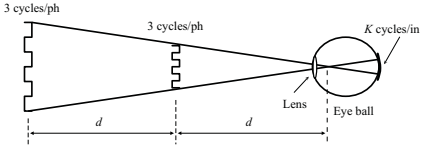


Spatial frequency units – solution

Instead, one commonly uses:

- cycles/ph (cycles per picture height) or
- cycles/deg (cycles per degree)

as the units of spatial frequencies, since both are distance-independent.



We will see such units when we discuss 2-D sinusoidal functions and human eye properties.