# Images and Displays

CS465 Lecture 2




## What is an image?

- A photographic print?
- A photographic negative?
- This projection screen?
- Some numbers in RAM?




## An image is:

- A 2D distribution of intensity or color
- A function defined on a two-dimensional plane

$$I : \mathbb{R}^2 \rightarrow \ldots$$

- Note: no mention of pixels yet
- To do graphics, must:
  - represent images—encode them numerically
  - display images—realize them as actual intensity distributions




## Representative display technologies

Computer displays

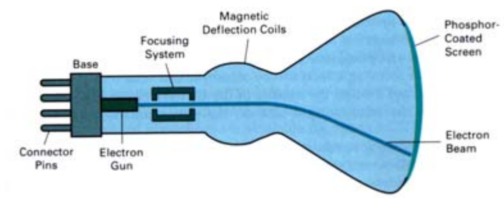
- Raster CRT display
- LCD display

Printers

- Laser printer
- Inkjet printer

## Cathode ray tube

- First widely used electronic display
  - developed for TV in the 1920s–1930s

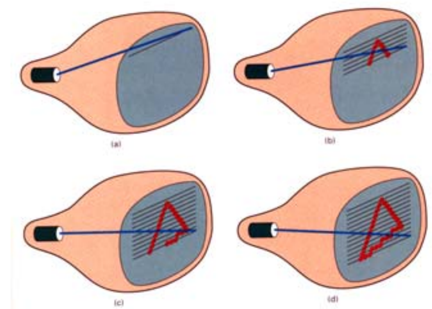



[H&B fig. 2-2]



## Raster CRT display

- Scan pattern fixed in display hardware
- Intensity modulated to produce image
- Originally for TV
  - (continuous analog signal)
- For computer, intensity determined by contents of *framebuffer*

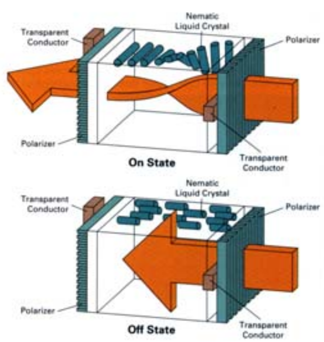



[H&B fig. 2-7]



## LCD flat panel or projection display

- Principle: block or transmit light by twisting its polarization
- Intermediate intensity levels possible by partial twist
- Fundamentally raster technology
- Fixed format

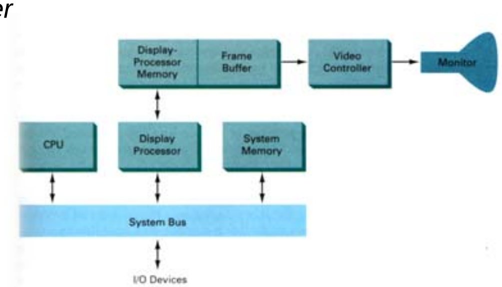



[H&B fig. 2-16]



## Raster display system

- Screen image defined by a 2D array in RAM
  - for CRT, read out and convert to analog in sync with scan
- In most systems today, it's in a separate memory
- The memory area that maps to the screen is called the *frame buffer*




[H&B fig. 2-29]

## Color displays

- Operating principle: humans are trichromatic
  - match any color with blend of three
  - therefore, problem reduces to producing 3 images and blending
- Additive color
  - blend images by sum
  - e.g. overlapping projection
  - e.g. unresolved dots
  - R, G, B make good primaries

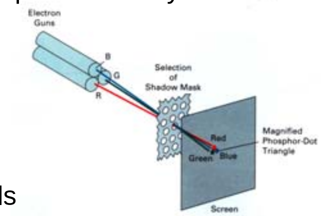

[cs417 S02 slides]





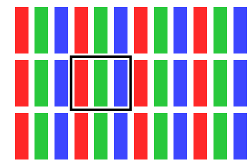
## Color displays

- CRT: phosphor dot pattern to produce finely interleaved color images




[H&B fig. 2-10]

- LCD: interleaved R,G,B pixels







## Laser printer

- Xerographic process
- Like a photocopier but with laser-scanned raster as source image
- Key characteristics
  - image is binary
  - resolution is high
  - very small, isolated dots are not possible

QuickTime™ and a TIFF (Uncompressed) decompressor are needed to see this picture.

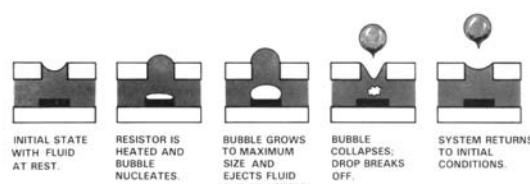
[howstuffworks.com]





## Inkjet printer

- Liquid ink sprayed in small drops
  - very small—measured in picoliters
- Head with many jets scans across paper
- Key characteristics:
  - image is binary (drop or no drop; no partial drops)
  - isolated dots are reproduced well




[cs417 S02 slides]

## Raster image representation

- All these devices suggest 2D arrays of numbers
- Big advantage: represent arbitrary images
  - approximate arbitrary functions with increasing resolution
  - works because memory is cheap (brute force approach!)

[Philip Greenspun]



## Meaning of a raster image

- Meaning of a given array is a function on 2D
- Define meaning of array = result of output device?
  - that is, piecewise constant for LCD, blurry for CRT
  - but: we don't have just one output device
  - but: want to define images we can't display (e.g. too big)
- Abstracting from device, problem is reconstruction
  - image is a sampled representation
  - pixel means "this is the intensity around here"
    - LCD: intensity is constant over square regions
    - CRT: intensity varies smoothly across pixel grid
  - will discuss specifics of reconstruction later



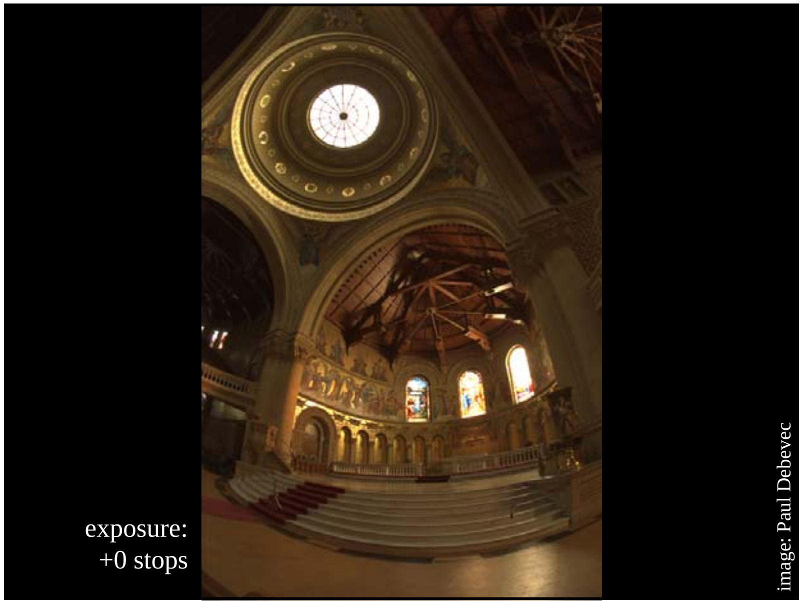
## Datatypes for raster images

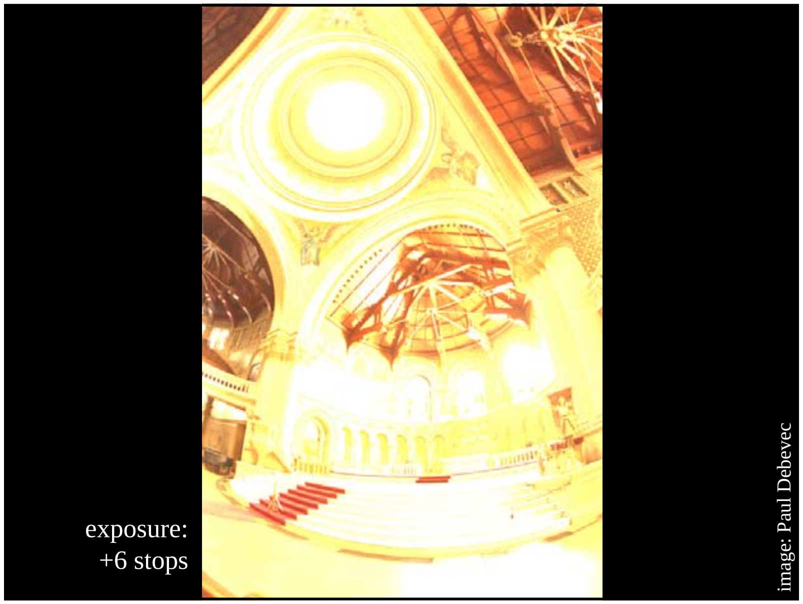
- Bitmaps: `boolean` per pixel (1 bpp): $I : \mathbb{R}^2 \to \{0, 1\}$
  - interp. = black and white; e.g. fax
- Grayscale: integer per pixel: $I : \mathbb{R}^2 \to [0, 1]$
  - interp. = shades of gray; e.g. black-and-white print
  - precision: usually `byte` (8 bpp); sometimes 10, 12, or 16 bpp
- Color: 3 integers per pixel: $I : \mathbb{R}^2 \to [0, 1]^3$
  - interp. = full range of displayable color; e.g. color print
  - precision: usually `byte[3]` (24 bpp)
  - sometimes 16 (5+6+5) or 30 or 36 or 48 bpp
  - indexed color: a fading idea



## Datatypes for raster images

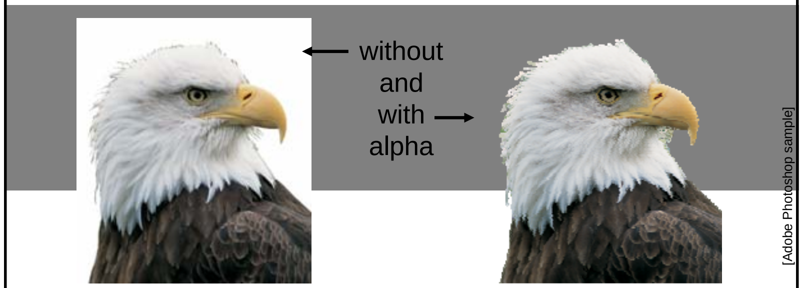
- Floating point: $I : \mathbb{R}^2 \to \mathbb{R}_+$ or $I : \mathbb{R}^2 \to \mathbb{R}_+^3$
  - more abstract, because no output device has infinite range
  - provides *high dynamic range* (HDR)
  - represent real scenes independent of display
  - becoming the standard intermediate format in graphics processors
- Clipping and white point
  - common to compute FP, then convert to integer
  - full range of values may not "fit" in display's output range
  - simplest solution: choose a maximum value, scale so that value becomes full intensity ($2^n$–1 in an *n*-bit integer image)


## Datatypes for raster images

- For color or grayscale, sometimes add *alpha* channel
  - describes transparency of images
  - more on this in a few lectures



## Storage requirements for images

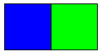
- 1024x1024 image (1 megapixel)
  - bitmap: 128KB
  - grayscale 8bpp: 1MB
  - grayscale 16bpp: 2MB
  - color 24bpp: 3MB
  - floating-point HDR color: 12MB



## Converting pixel formats

- Color to gray
  - could take one channel (blue, say)
    - leads to odd choices of gray value
  - combination of channels is better
    - but different colors contribute differently to lightness
    - which is lighter, full blue or full green?
    - good choice: gray = 0.2 R + 0.7 G + 0.1 B
    - more on this in color, later on

Same pixel values.



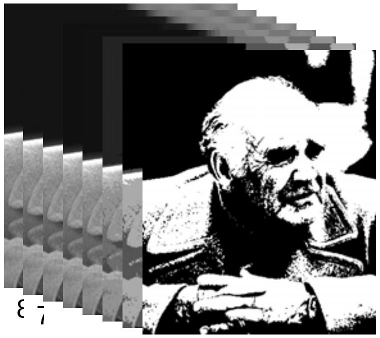
Same luminance?






## Converting pixel precision

- Up is easy; down loses information—be careful



1 bpp (2 grays)

[Philip Greenspun]



## Dithering

- When decreasing bpp, we quantize
- Make choices consistently: banding
- Instead, be inconsistent—dither
  - turn on some pixels but not others in gray regions
  - a way of trading spatial for tonal resolution
  - choose pattern based on output device
  - laser, offset: clumped dots required (halftone)
  - inkjet, screen: dispersed dots can be used

## Dithering methods

- Ordered dither
  - based on traditional, optically produced halftones
  - produces larger dots
- Diffusion dither
  - takes advantage of devices that can reproduce isolated dots
  - the modern winner for desktop printing

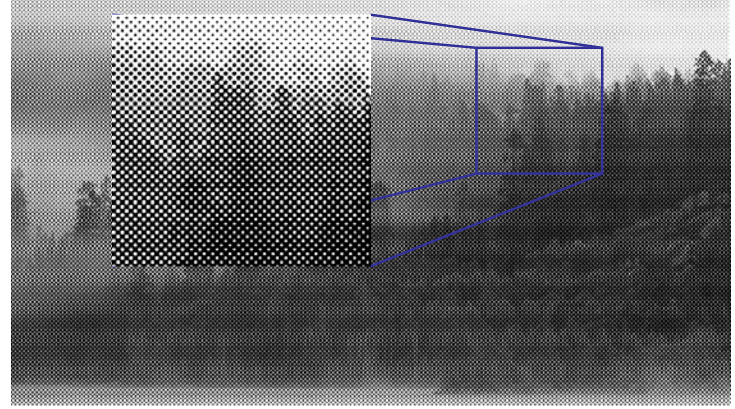
[Philip Greenspun]





## Ordered Dither example

- Produces regular grid of compact dots



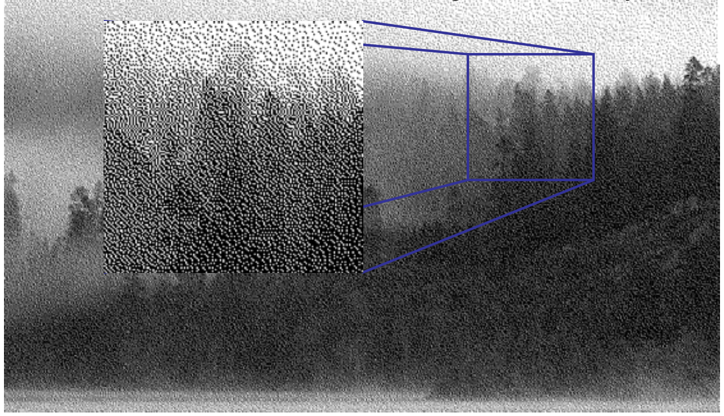
[Philip Greenspun]





## Diffusion dither

- Produces scattered dots with the right local density



[Philip Greenspun]





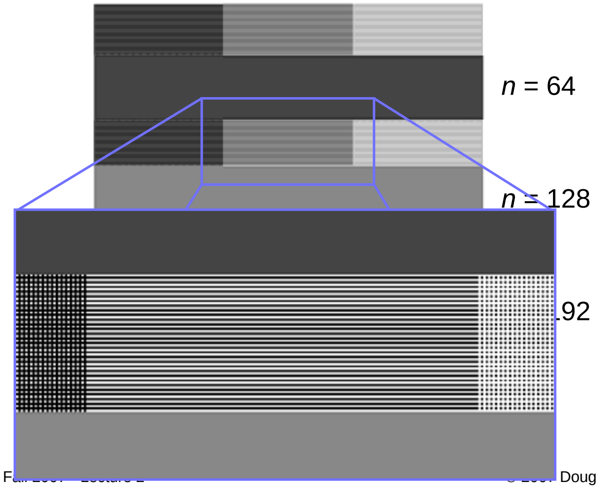
## Intensity encoding in images

- What do the numbers in images (pixel values) mean?
  - they determine how bright that pixel is
  - bigger numbers are (usually) brighter
- *Transfer function*: function that maps input pixel value to luminance of displayed image

$$I = f(n) \quad f : [0, N] \rightarrow [I_{\min}, I_{\max}]$$

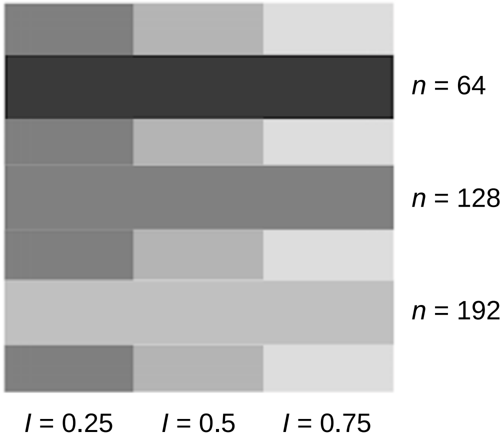
- What determines this function?
  - physical constraints of device or medium
  - desired visual characteristics

## What this projector does





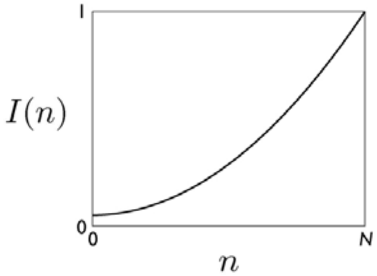

## What this projector does (simulated)






## What this projector does

- Something like this:






## Constraints on transfer function

- Maximum displayable intensity, $I_{max}$
  - how much power can be channeled into a pixel?
    - LCD: backlight intensity, transmission efficiency (<10%)
    - projector: lamp power, efficiency of imager and optics
- Minimum displayable intensity, $I_{min}$
  - light emitted by the display in its “off” state
    - *e.g.* stray electron flux in CRT, polarizer quality in LCD
- Viewing flare, *k*: light reflected by the display
  - very important factor determining image contrast in practice
    - 5% of $I_{max}$ is typical in a normal office environment [sRGB spec]
    - much effort to make very black CRT and LCD screens
    - all-black decor in movie theaters

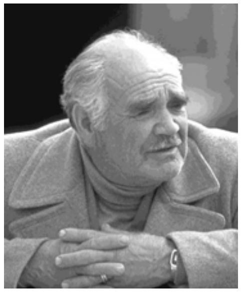
## Dynamic range

- Dynamic range $R_d = I_{max} / I_{min}$ , or $(I_{max} + k) / (I_{min} + k)$
  - determines the degree of image contrast that can be achieved
  - a major factor in image quality
- Ballpark values
  - Desktop display in typical conditions: 20:1
  - Photographic print: 30:1
  - Desktop display in good conditions: 100:1
  - Photographic transparency (directly viewed): 1000:1
  - High dynamic range display: 10,000:1



## Transfer function shape

- Desirable property: the change from one pixel value to the next highest pixel value should not produce a visible contrast
  - otherwise smooth areas of images will show visible bands
- What contrasts are visible?
  - rule of thumb: under good conditions we can notice a 2% change in intensity
  - therefore we generally need smaller quantization steps in the darker tones than in the lighter tones
  - most efficient quantization is logarithmic



An image with severe *banding*



## How many levels are needed?

- Depends on dynamic range
  - 2% steps are most efficient:

    $$0 \mapsto I_{\min}; 1 \mapsto 1.02 I_{\min}; 2 \mapsto (1.02)^2 I_{\min}; \ldots$$

  - log 1.02 is about 1/120, so 120 steps per decade of dynamic range
    - 240 for desktop display
    - 360 to print to film
    - 480 to drive HDR display
- If we want to use linear quantization (equal steps)
  - one step must be < 2% (1/50) of $I_{min}$
  - need to get from ~0 to $I_{min} \cdot R_d$ so need about 50 $R_d$ levels
    - 1500 for a print; 5000 for desktop display; 500,000 for HDR display
- Moral: 8 bits is just barely enough for low-end applications
  - but only if we are careful about quantization



## Intensity quantization in practice

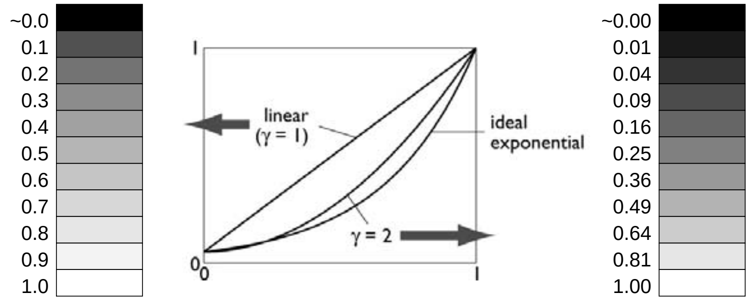
- Option 1: linear quantization $I(n) = (n/N)\, I_{\max}$
  - pro: simple, convenient, amenable to arithmetic
  - con: requires more steps (wastes memory)
  - need 12 bits for any useful purpose; more than 16 for HDR
- Option 2: power-law quantization $I(n) = (n/N)^{\gamma}\, I_{\max}$
  - pro: fairly simple, approximates ideal exponential quantization
  - con: need to linearize before doing pixel arithmetic
  - con: need to agree on exponent
  - 8 bits are OK for many applications; 12 for more critical ones
- Option 2: floating-point quantization $I(x) = (x/w)\, I_{\max}$
  - pro: close to exponential; no parameters; amenable to arithmetic
  - con: takes more than 8 bits
  - 16–bit “half precision” format is becoming popular

## Why gamma?

- Power-law quantization, or *gamma correction* is most popular
- Original reason: CRTs are like that
  - intensity on screen is proportional to voltage$^2$
- Continuing reason: inertia + memory savings
  - inertia: gamma correction is close enough to logarithmic that there's no sense in changing
  - memory: gamma correction makes 8 bits per pixel an acceptable option



## Gamma quantization




- Close enough to ideal perceptually uniform exponential



## Gamma correction

- Sometimes (often, in graphics) we have computed intensities *a* that we want to display linearly
- In the case of an ideal monitor with zero black level,

  $$I(n) = (n/N)^\gamma$$

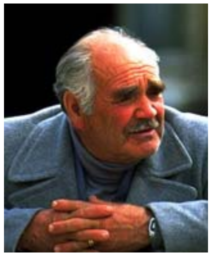
  (where N = $2^n - 1$ in *n* bits). Solving for *n*:

  $$n = N a^{\frac{1}{\gamma}}$$

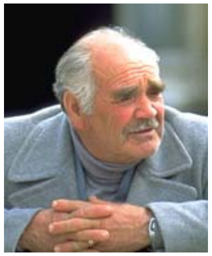
- This is the "gamma correction" recipe that has to be applied when computed values are converted to 8 bits for output
  - failing to do this (implicitly assuming gamma = 1) results in dark, oversaturated images



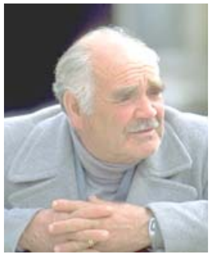
## Gamma correction



corrected for γ lower than display



OK



corrected for γ higher than display

[Philip Greenspun]