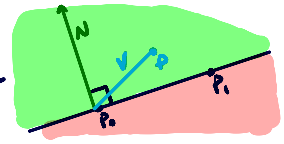
CS 184 Midterm 1 Study Guide — Jeffrey Shen

① **Computer Graphics:** use of computers to synthesize and manipulate visual information

② **Rasterization** drawing a triangle to the framebuffer, pts → pixels

1) Sample if pixel center inside triangle

Line Equation: test if point inside triangle
- Use 3 line test to see if inside triangle



$L(x,y) = V \cdot N = -(x-x_0)(y_1-y_0) + (y-y_0)(x_1-x_0)$

$L(x,y) > 0$

$L(x,y) < 0$

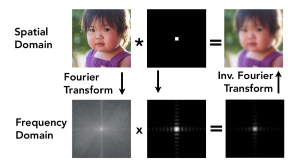
③ **Sampling & Aliasing**

Aliasing: artifacts from sampling, sample fast signals too slowly, jaggies, wagon wheel effect

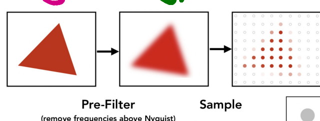
Antialias by 1) increase sampling rate 2) filter signal frequences above Nyquist freq

Nyquist Theorem: no aliasing from frequencies in signal less than Nyquist frequency

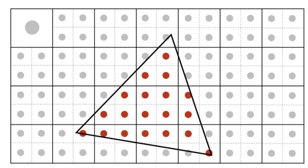
Nyquist Freq: $f_{Nyquist} = \frac{1}{2} f_{sampling}$ Nyquist Rate: $f_{sampling} > 2 f_{signal}$



2) Filter signals above Nyquist



3) Supersample, approx 1 pixel box filter by sampling N×N locations within pixel and avg



④ **Transforms** functions acting on pts

Linear Transforms: matrices $x' = Mx$ $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

a) rotation

$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

b) Scale

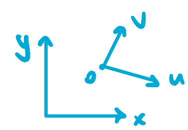
$S_{(s_x,s_y)} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$

Affine Transforms: linear + translation

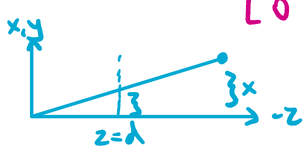
$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

- compose transformations by matrix multiplication but not commutative

Coordinate System Transform $F = \begin{bmatrix} u_x & v_x & o_x \\ u_y & v_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$



Projective Transforms:



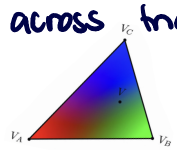
$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} x \cdot d/z \\ y \cdot d/z \\ d \end{pmatrix}$

⑤ **Texture Mapping** Each surface pt assigned texture coordinate

Barycentric Coordinates: interpolating across triangles

$(x,y) = \alpha A + \beta B + \gamma C$

$\alpha + \beta + \gamma = 1$



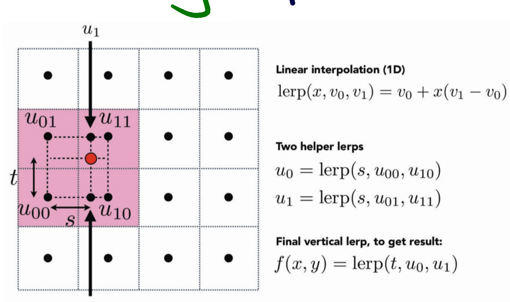
$$\alpha = \frac{-(x - x_B)(y_C - y_B) + (y - y_B)(x_C - x_B)}{-(x_A - x_B)(y_C - y_B) + (y_A - y_B)(x_C - x_B)}$$

$$\beta = \frac{-(x - x_C)(y_A - y_C) + (y - y_C)(x_A - x_C)}{-(x_B - x_C)(y_A - y_C) + (y_B - y_C)(x_A - x_C)}$$

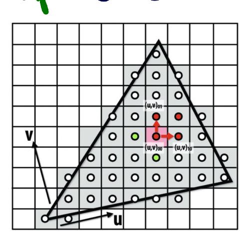
$$\gamma = 1 - \alpha - \beta$$

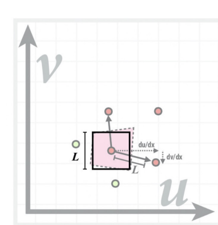
- Can take Jacobian of (1,0), (0,1) in u,v space to find footprint texture area

Bilinear Filtering: Sample texture value from pixels



Mipmap: store lower resolution texture images





mipmap level D

$$D = \log_2 L$$

$$L = \max\left(\sqrt{\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2}, \sqrt{\left(\frac{du}{dy}\right)^2 + \left(\frac{dv}{dy}\right)^2}\right)$$

⑥ Raserization Pipeline

vertex processing → triangle → raserization → fragment → framebuffer

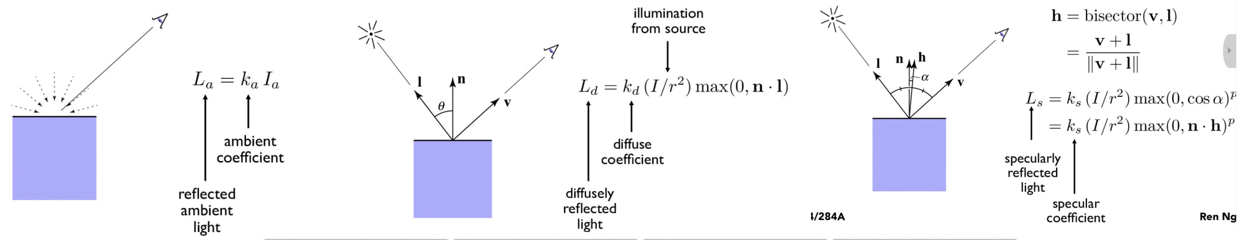
Z-buffer: hidden surface removal algorithm: Store min z value and track depth buffer

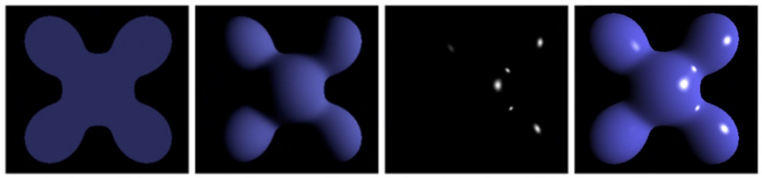
Blinn-Phong Reflection Model: local shading model, per pixel

Ambient Shading constant color & shadows

Diffuse Shading independent of view direction

Specular Shading Intensity depends on view dir



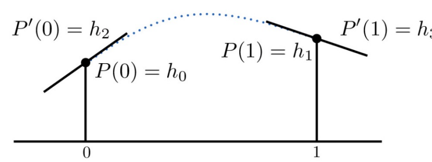


Ambient + Diffuse + Specular = Phong Reflection

⑦ Splines, Bezier Curves

Cubic Hermite Interpolation: Given values and derivatives as input, give cubic poly that interpolates, $P(x) = at^3 + bt^2 + ct + d$

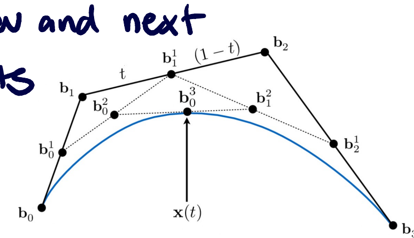
$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix}$$



Catmull-Rom: to calculate derivatives, match slope between prev and next

Bézier Curves: draw spline using points by creating tangents

de Casteljau Algorithm: insert points using linear interpolation, repeat recursively $b_0^1(t) = (1-t)b_0 + t b_1$

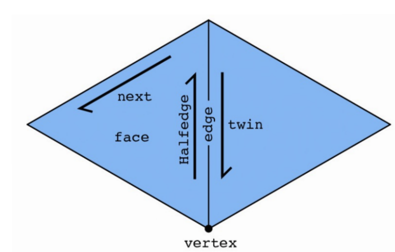


Bézier Surfaces: extend curves to surfaces (u,v)
- evaluate de Casteljau using point u, eval v on moving curve

⑧ Mesh Representations & Geometry

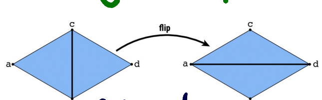
Mesh Representations: List of triangles, points, half-edge structure

Half-edge Data Structure: two half-edges act as "glue" between mesh elements
- use twin and next to traverse, process



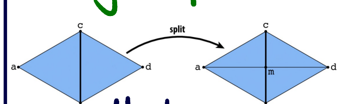
Operations

Edge Flip



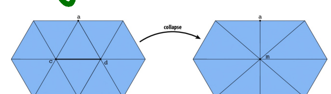
resample

Edge Split



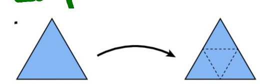
add elements

Edge Collapse



delete elements

Loop Subdivision



1) Split all
2) Flip new edges

Assigning new vertex weights, adding elements by touching new & old

Catmull-Clark Subdivision: Add vertex in face, add midpoint on each edge, connect
- can use special processing for sharp creases, marking them as sharp
- downsample by collapsing edges, choose lowest quadric error

Quadric Error: approx distance to surface as sum of distances to planes

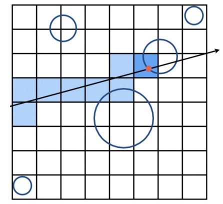
Mesh Regularization: flip if degree large, center vertices

⑨⑩ Ray Tracing
light rays for photorealistic rendering

Recursive Ray Tracing: bounce from one object to another until max level or non specular

Ray Surface Intersection:

ray: $r(t) = O + td$ (O ← origin, d ← unit direction)   plane: $(p - p') \cdot N = 0$ intersection: $t = \frac{(p' - O) \cdot N}{d \cdot N}$

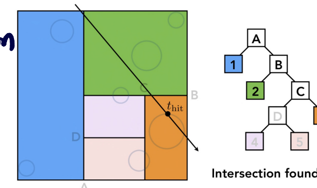


- Use bounding volumes to check general intersections

Uniform Spatial Partitions: find bounding box, create grid, store in overlapping cells
- use right grid resolutions

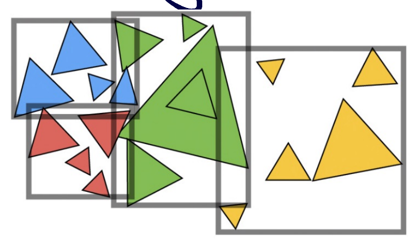
Non-Uniform Spatial Partitions: ex) KD-trees want to min ray intersection

1) Find bounding box 2) split cells until termination
3) Store obj references in leaf nodes



Object Partition: Partition set of objects into disjoint subsets, ex) Bounding Vol Hier (BVH)

1) Find bounding box 2) Recursively split set into two subsets
3) Stop when few obj

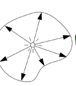


Surface Area Heuristic: prob of hitting shape is ratio of SA

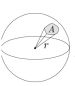
⑪ Radiometry & Photometry
measurement system for illumination
photometry limited to eye

Radiant (luminous) Energy: Q [J = Joule] energy of electromagnetic radiation

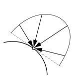
Radiant flux: $\Phi = \frac{\partial Q}{\partial t}$ [W = Watt][lm = lumen] energy emitted, reflected, transmitted per time



Radiant Intensity: $I(\omega) = \frac{\partial \Phi}{\partial \omega}$ $\left[\frac{W}{sr}\right]\left[\frac{lm}{sr} = cd = \text{candela}\right]$ power per unit solid angle emitted



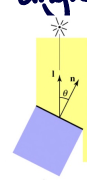
Solid Angle: $\Omega = \frac{A}{r^2}$, sphere 4π steradians, ratio of subtended area on sphere to r^2



Irradiance: $E(x) = \frac{\partial \Phi(x)}{\partial A}$ $\left[\frac{W}{m^2}\right]\left[\frac{lm}{m^2} = lux\right]$ power per unit area incident on surface point

Radiance: $L(p,\omega) = \frac{d^2 \Phi(p,\omega)}{\partial\omega\, \partial A \cos\theta}$ $\left[\frac{W}{sr\, m^2}\right]\left[\frac{cd}{m^2} = \frac{lm}{sr\, m^2} = nit\right]$ power emitted, transmitted, by surface per unit solid angle, per projected area

Lambert's Cosine Law: Irradiance at surface is proportional to cosine of angle between light and surface normal

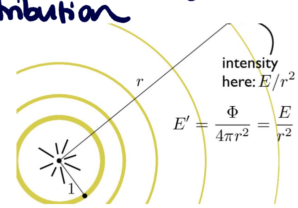


In general, power per unit area is proportional to $\cos\theta = l \cdot n$

$$E = \frac{\Phi}{A}\cos\theta$$

Irradiance Falloff: if emitting Φ in uniform angular distribution

$$E' = \frac{\Phi}{4\pi r^2}$$



⑫ Monte Carlo Integration

Monte Carlo Integration: estimate integral based on random sampling of function

↑ works for general functions, high dimensions

↓ noise, slow to converge

For function $f(x)$, RV $X_i \sim p(x)$

$$F_N = \frac{1}{N}\sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)}$$

$X \sim p(x)$

$p(x) \geq 0$ $\int p(x)\,dx = 1$

$E[X] = \int x\, p(x)\, dx$

Unbiased Estimator: if expected value of estimator is desired integral
- More samples reduces variance

For direct lighting estimate
compute L_i at p from ω_i, increment Monte Carlo $F_N := F_N + \frac{2\pi}{N} L_i \cos\theta_i$

Importance Sampling: $\frac{1}{n}\sum_{i=1}^{n} \frac{f_i(x_i)}{p(x_i)}$

Inversion Method: draw samples from probability distribution
1) calculate pdf $p(x) = cf(x)$ 2) calculate cdf $P(x) = \int p(x)dx$ 3) solve $x = P^{-1}(\xi)$

Inversion Ex)

1) $f(x) = x^2$
$p(x) = cf(x)$
$\int_0^2 cf(x) = 1$
$c = 3/8$
$p(x) = \frac{3}{8}x^2$

2) $P(x) = \int \frac{3}{8}x^2\,dx$
$P(x) = \frac{x^3}{8}$

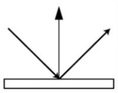
3) $x = \sqrt[3]{8\xi}$

(13) Global Illumination & Path Tracing

Reflection: light incident on surface leaves surface on incident w/o change in freq

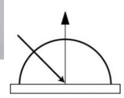
1) Ideal Specular

mirror



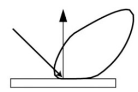
2) Ideal Diffuse

all dir

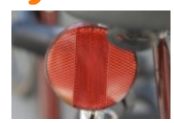


3) Glossy Specular

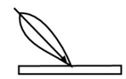
majority near mirror



4) Retro-reflective



reflected back toward source



Bidirectional reflectance distribution function (BRDF): how much light reflected into each outgoing direction from incoming direction

$$f_r(\omega_i \to \omega_r) = \frac{\partial L_r(\omega_r)}{\partial E_i(\omega_i)} = \frac{\partial L_r(\omega_r)}{L_i(\omega_i)\cos\theta_i\, d\omega_i} \left[\frac{1}{sr}\right]$$

Reflection Equation: $L_r(p,\omega_r) = \int_{H^2} f_r(p,\omega_i \to \omega_r) L_i(p,\omega_i) \cos\theta \, d\omega_i$

Rendering Equation: $L_o(p,\omega_o) = L_e(p,\omega_o) + \int_{H_2} f_r(p,\omega_i \to \omega_o) L_o(tr(p,\omega), -\omega_i) \cos\theta_i \, d\omega_i$

$L_o = L_e + K(L_o)$, $K = R \circ T$

$L = (I - K)^{-1}(L_e)$

Global Illumination: Russian Roulette
probabilistic termination of recursion to get sum of all paths w/o infinite

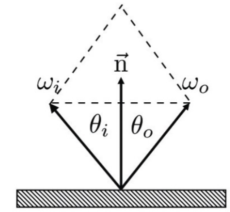
CS 184 Midterm 2 Study Guide

14-21

(14) Material Modeling

Material == BRDF

Perfect Specular Reflection: $\omega_o = -\omega_i + 2(\omega_i \cdot \vec{n})\vec{n}$

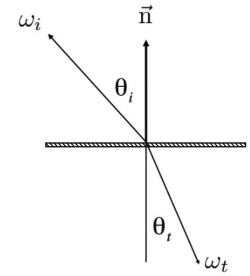


$$\theta = \theta_o = \theta_i$$

Specular Refraction: light can be transmitted through surface when it enters a new medium

Snell's Law: transmitted angle depends on index of refraction (IOR) for incident ray, index of reflection (IOR) for exiting

$$\cos\theta_t = \sqrt{1 - \left(\frac{\eta_i}{\eta_t}\right)^2 (1 - \cos^2\theta_i)}$$



$$\eta_i \sin\theta_i = \eta_t \sin\theta_t$$

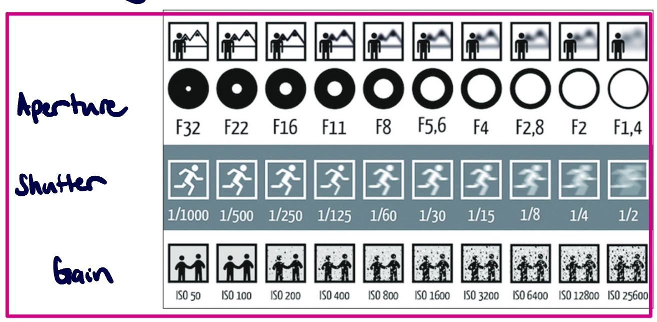
Microfacet Material Model: concentrated distributed microfacet normals ⇒ glossy
spread out normals ⇒ diffuse

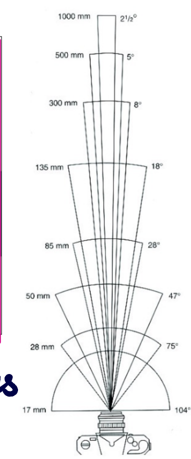
Anisotropic Materials: directionality of underlying surface

(15/16) Cameras & Lenses

sensor accumulates irradiance

Aperture Size: change f-stop by opening or closing aperture





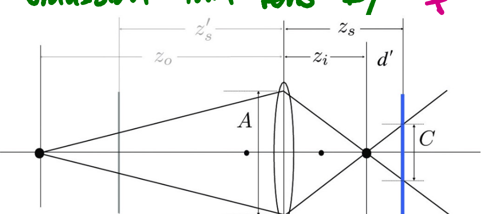
Shutter Speed: Change the duration the sensor pixels integrate light

ISO: change the amplification between sensor values and digital image values

Exposure: time × irradiance $Q = T \times E$

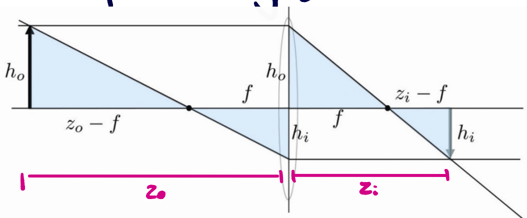
Thin Lens Approximation: Assume all parallel rays entering a lens pass through its focal pt

Gaussian Thin Lens Eq: $\frac{1}{f} = \frac{1}{z_i} + \frac{1}{z_o}$



$$\frac{C}{A} = \frac{d'}{z_i} = \frac{|z_s - z_i|}{z_i}$$

f number $N = \frac{f}{D}$ — f: focal length, D: diameter

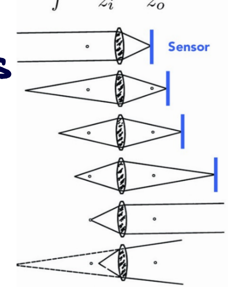


$$\frac{h_o}{z_o - f} = \frac{h_i}{f} \qquad \frac{h_o}{f} = \frac{h_i}{z_i - f}$$

Depth of Field: range of object depths that are rendered w acceptable sharpness

Contrast Detection Autofocus: high contrast when in focus, adjust focus until contrast max

$$\frac{1}{f} = \frac{1}{z_i} + \frac{1}{z_o}$$



- z_i and z_o are called conjugate points
- To focus on objects at different distances, move the sensor relative to the lens
- For $z_i < z_o$ the object is larger than the image
- At $z_i = z_o$ we have 1:1 macro imaging
- For $z_i > z_o$ the image is larger than the object (magnified)
- Can't focus on objects closer than the lens' focal length

(17) Intro to Animation

Animation Principles

1) Squash and Stretch
2) Anticipation
3) Staging
4) Follow Through
5) Ease-In & Ease-Out
6) Arcs
7) Secondary Action
8) Timing
9) Exaggeration
10) Appeal
11) Personality

Keyframe: motions, vector of parameter values

Forward Kinematics: animator provides angles, computer determines position

Inverse Kinematics: Given end position, find joint angles

Hard: multiple or no solutions

Kinematics: ↑ direct control is convenient, straight forward implementation
↓ could be inconsistent w/ physics, time consuming

Skinning: move the surface along w/ bones, blend together linearly

Linear Blend Skinning (LBS): $v' = \sum_{j \in H} w_j(v) T_j \begin{pmatrix} v \\ 1 \end{pmatrix}$ (w_j ← influence, T_j ← transformation)

Rigging: controls for character

Motion Capture: record real world performances, extract pose from data
↑ capture large amounts of data quickly, realism
↓ complex and costly

(18) Intro to Physical Simulation

Newton's Law: $F = ma$

Mass & Spring Systems

Spring: $f_{a \to b} = k_s \frac{b-a}{\|b-a\|}(\|b-a\| - 1)$ w damping: $f_a = -k_d \frac{b-a}{\|b-a\|}(\dot{b} - \dot{a}) \cdot \frac{b-a}{\|b-a\|}$

Euler's Method (Forward Euler / Explicit): Simple iterative method

$$x^{t+\Delta t} = x^t + \Delta t\, \dot{x}^t$$
$$\dot{x}^{t+\Delta t} = \dot{x}^t + \Delta t\, \ddot{x}^t$$

Implicit:

$$x^{t+\Delta t} = x^t + \Delta t\, \dot{x}^{t+\Delta t}$$
$$\dot{x}^{t+\Delta t} = \dot{x}^t + \Delta t\, \ddot{x}^{t+\Delta t}$$

Verlet Integration: constrain positions to prevent unstable behavior

Particle Physics: Gravitational Attraction: $F_g = G \frac{m_1 m_2}{d^2}$

(19/20) Intro to Color Science

- Color Perception is Highly Adaptive: Chromatic Adaptation
- Want to match color on display w/ real world

Color: visual sensations that arise from seeing light of different spectral distribution

Spectral Power Distribution (SPD): amount of light present at each wavelength
radiometric units/nm (watts/nm)

Tristimulus Theory of Color: can match every color w/ either 3 primary colors, or 2 colors and subtract color from .. test

Metamer: two spectra (∞-dim) that project to same (S,M,L) response

Color reproduction: at each pixel, choose R,G,B values so output matches target
- choose value so s' in display project to same eye SML response

Color perceived for display spectra with values R,G,B

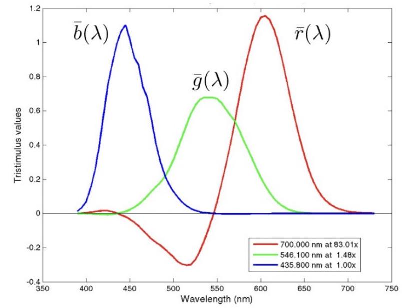
$$\begin{bmatrix} S \\ M \\ L \end{bmatrix}_{disp} = \begin{bmatrix} - & r_S & - \\ - & r_M & - \\ - & r_L & - \end{bmatrix} \begin{bmatrix} | & | & | \\ s_R & s_G & s_B \\ | & | & | \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Color perceived for real scene spectra, s

$$\begin{bmatrix} S \\ M \\ L \end{bmatrix}_{real} = \begin{bmatrix} - & r_S & - \\ - & r_M & - \\ - & r_L & - \end{bmatrix} \begin{bmatrix} | \\ s \\ | \end{bmatrix}$$

Solution (form #3):

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \underbrace{\begin{bmatrix} r_S \cdot s_R & r_S \cdot s_G & r_S \cdot s_B \\ r_M \cdot s_R & r_M \cdot s_G & r_M \cdot s_B \\ r_L \cdot s_R & r_L \cdot s_G & r_L \cdot s_B \end{bmatrix}^{-1} \begin{bmatrix} - & r_S & - \\ - & r_M & - \\ - & r_L & - \end{bmatrix}}_{3xN} \begin{bmatrix} | \\ s \\ | \end{bmatrix}$$



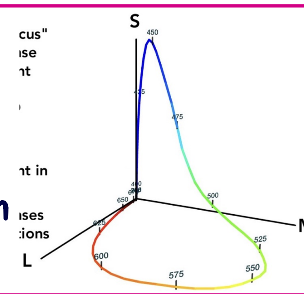
Color Gamuts

1) Standardized RGB (sRGB)
2) CIE XYZ, XYZ span all observable colors, Y luminance

Luminance: integral radiance by luminous efficiency

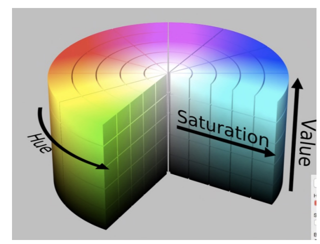
$$Y = \int \Phi(\lambda) V(\lambda) \, d\lambda$$

plot of S,M,L response functions as a pt in 3D space



Chromaticity: x,y,z as $x = \frac{X}{X+Y+Z}$ $y = \frac{Y}{X+Y+Z}$ $z = \frac{Z}{X+Y+Z}$

HSV: color space perceptually organized, hue: kind, saturation: colorfulness
value: lightness



21) Image Sensors

CMOS APS: memory laid out in 2d array

Quantum Efficiency: $QE = \frac{\#\text{electrons}}{\#\text{photons}}$

High Dynamic Range (HDR) through multiple exposures

- Most cameras are Backside Illumination (BSI) - higher QE, lower cross-talk

Signal-to-Noise Ratio (SNR): $SNR = \frac{\text{mean pixel value}}{\text{std dev of pixel value}} = \frac{\mu}{\sigma}$ $SNR(dB) = 20\log_{10}\left(\frac{\mu}{\sigma}\right)$

Use Poisson to model # photons arriving in exposure so $SNR = \frac{\mu}{\sigma} = \sqrt{\lambda}$ λ: photons

Pixel Noise:
1) Dark Current: electrons disloged by thermal activity
2) Hot Pixels: leaking in due to manufacturing defects
3) Read Noise: Thermal noise in readout circuitry

22) Image Processing

JPEG Compression: convert to Y'CbCr color space Y' (lightness), Cr/Cb (Chroma colors)
Compression in CbCr bc insensitive to color errors
Discrete Cosine Transform (DCT), use quantization to reduce dimensionality

Basic Image Processing: blur, sharpen, edge detection

Convolution: $(f * I)(x,y) = \sum_{i,j=-\infty}^{\infty} f(i,j) I(x-i, y-j)$

Gaussian Blur: $f(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}$

Convolve by frequency domain, then convert back

Data-Dependent Filters:

Median: Replace pixel w/ median of neighbors

Bilateral: output is weighted sum of pixels in support region, combination of spatial distance and intensity difference

Denoise by non-local means:

search for similar neighborhood

Non-Parametric Texture Synthesis:

find prob func for patches that are similar

(23) Light Field Cameras

4D Light Fields Capture radiance flowing along every ray

Light Field Camera capture light field flowing into lens in every shot

Light Field Sensor: microlens array in front of sensor

Can use computational refocusing and lens aberration correction