THE DIGITAL IMAGE

An image is a continuous function

$$f: \mathbb{R}^n \to S$$

that can be discretized to

$$I: [X] \times [Y] \to S$$

by sampling at a sufficiently high rate. A pixel represents one sample.

Nyquist Frequency

1/2 of the discrete sampling frequency. Signals at or above it cannot be reconstructed perfectly from their sampled representation.

Quantization

A digital image

$$I_d: [X] \times [Y] \to [Q]^3$$

has a discrete codomain. The function g, s.t. $I_d = g \circ I$, performs quantization and is always lossy.

Resolution

image size (in pixels), geometric (pixels per area), radiometric (quantization density)

Noise

additive $I(x,y) = f(x,y) + c$

Gaussian [PDF $c \mapsto (\sigma\sqrt{2\pi})^{-1} \exp(-\frac{1}{2}(\frac{c}{\sigma})^2)$]

Poisson [PDF $c \mapsto \frac{\lambda^c \exp(-\lambda)}{k!}$]

Rician

multiplicative $I(x,y) = c f(x,y)$

Quantization errors

signal-to-noise ratio (SNR)

$$\frac{1}{|X||Y|} \sum_{x \in [X]} \sum_{y \in [Y]} f(x,y)$$

peak signal-to-noise ratio (PSNR)

$$\Leftarrow \max_{(x,y) \in [X] \times [Y]} f(x,y)$$

Camera

CCD ⊖ high power, blooming, smearing, darkcurrent

CMOS ⊖ rolling shutter, more sensor noise

colors via prism, mosaic, or wheel

COLOR

Humans perceive colors in 3D, so to identify all differentiable colors uniquely, one needs 3 primary colors.

CIE XYZ Color Space

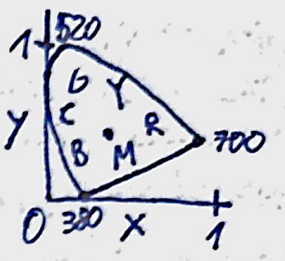
Based on the color matching experiment, synthesize wavelength response curves $\bar{x}, \bar{y}, \bar{z}$ and define $(X,Y,Z) := (\int P(\lambda)\bar{x}(\lambda)d\lambda, \ldots \bar{y} \ldots, \ldots \bar{z} \ldots)$.

Chromaticity $(x,y) = (\frac{X}{X+Y+Z}, \frac{Y}{X+Y+Z})$ and brightness Y.

resp.

Chromaticity Chart

dominant wavelength, inverse color, white point at $(1/3, 1/3)$



Other Color Spaces

sRGB, CMY(K) with $\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = 1 - \begin{pmatrix} R \\ G \\ B \end{pmatrix}$, YIQ (luminance, in-phase orange/blue, quadrature purple/green), HSV, HSL/HSB

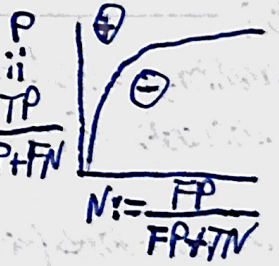
IMAGE SEGMENTATION

Partition an image I into regions $R_1, \ldots, R_n$ s.t. $I = \bigcup R_i$ with $R_i \cap R_j = \emptyset$ for $i \neq j$ for some specified use case.

Thresholding

Set TER using trial/error, ground truth comparison, or ROC curve optimization. Then, define $R_0 := \{p \in [X] \times [Y] \mid I(p) < T\}$ and $R_1 := I \setminus R_0$.

ROC-curve



plot sensitivity and specificity.

[optimal / worst

optimal operating mode is where the gradient of ROC is $\frac{N(V_{TN}+C_{FP})}{P(V_{TP}+C_{FN})}$.

Region Growing

Refine a segmentation by growing regions from seeds using domain knowledge (expected shapes), distribution models etc.

Chroma keying

In a studio, use a background of known color (green/blue) and perform thresholding on Euclidean color distance $\|I-g\| \geq T$. Careful with subject's color, hard alpha mask, motion blur etc.

To improve quality, sample N background points and calculate their mean μ and covariance matrix $\Sigma := \frac{\Sigma(\Omega_i - \mu)^T(\Omega_i-\mu)}{N-1}$.

Then, decide using the Mahalanobis distance $\sqrt{(x-\mu)^T\Sigma^{-1}(x-\mu)}$.

Plain Background Subtraction

From a video, compute per-pixel model with μ and Σ across time. Then, again, use the Mahalanobis distance.

Morphological Operators

Logical pixel transformations based on comparison of neighborhoods with a pattern:

EROSION: keep iff. kernel matches perfectly
DILATION: keep iff. kernel matches ≥ 1 pixels
OPENING: erode $\Rightarrow$ dilate (remove noise)
CLOSING: dilate $\Rightarrow$ erode (Fill holes)

FILTERING

Functions that given an $X \times Y$ image produce another $X \times Y$ image are filters.

Linear Filters

L is linear $\Leftrightarrow L(\alpha I_1 + \beta I_2) = \alpha L(I_1) + \beta L(I_2)$
$\Leftrightarrow L(I)(x,y) = \sum_{(i,j)\in N} K(x,y;i,j)\, I(x-i, x-j)$

Shift-Invariant Linear Filters

L is LSI $\Leftrightarrow$ L is linear and weights are independent of (x,y)
$\Leftrightarrow L(I)(x,y) = \sum_{(i,j)\in N} K(i,j)\, I(x-i, x-j)$

Correlation

$$(K \circ I)(x,y) = \sum_{i=-k}^{k}\sum_{j=-k}^{k} K(i,j)\, I(x+i, y+j)$$

useful to measure similarity 

Convolution

$$(K * I)(x,y) = \sum_{i=-k}^{k}\sum_{j=-k}^{k} K(i,j)\, I(x-i, y-j)$$

useful to point-spread 

Convolution is correlation with a flipped kernel, i.e. $K * I = (K^T) \circ I$.

Separable Kernels

If f, g exist, s.t. $K(m,n) = f(m)\cdot g(n)$, then calculating a pixel is $O(k)$ versus $O(k^2)$. Two 1D-passes replace one 1D-pass.

Gaussian Kernel

$G_\sigma(x,y) = (2\pi\sigma^2)^{-1}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$

Self-convolution yields Gaussian: $G_\sigma * G_\sigma = G_{\sqrt{2}\sigma}$

Rotationally symmetric, single lobe in space and frequency domain, separable.

Differential Kernels

$\frac{\partial}{\partial x} I$ $[-1\ 1]$, Prewitt $\begin{bmatrix}-1&0&1\\-1&0&1\\-1&0&1\end{bmatrix}$,

Sobel $\begin{bmatrix}-1&0&1\\-2&0&2\\-1&0&1\end{bmatrix}$, Laplace $\begin{bmatrix}0&1&0\\1&-4&1\\0&1&0\end{bmatrix}$,

high-pass $\begin{bmatrix}-1&-1&-1\\-1&8&-1\\-1&-1&-1\end{bmatrix}$. All sum up to 0.

EDGE DETECTION

Derivative, Prewitt, or Sobel with thresholding.

Better: Detect zero-crossings of 2nd derivative but blur first:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left(1-\frac{x^2+y^2}{2\sigma^2}\right)\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

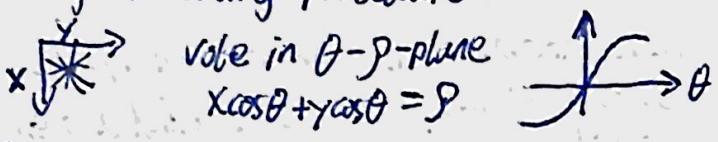
Even Better:

Canny Edge Detection

1. Smoothen image with Gaussian.
2. Compute gradient magnitude and quantized $(|, \backslash, -, /)$ edge direction.
3. Apply nonmaxima suppression to magnitude image based on neighbors in correct direction.
4. Double-threshold weak and strong pixels.
5. Region-grow with all strongs as seeds.

Hough Transform

Given a per-pixel edge predicate, try to find lines, circles, or other curves globally through a voting procedure.



vote in θ-ρ-plane

$x\cos\theta + y\cos\theta = \rho$

CORNER DETECTION

Find points (x,y) where local displacement sensitivity $S(\Delta x, \Delta y) = \sum_{x=-k}^{k}\sum_{y=-k}^{k}\left(f(x,y) - f(x+\Delta x, y+\Delta y)\right)^2$ is maximized. Fixing $\Delta x, \Delta y$ s.t. $\left\|\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix}\right\| = 1$ allows for the approximation

$$S(\Delta x, \Delta y) = \sum_{x=-k}^{k}\sum_{y=-k}^{k}\left(\left(f_x(x,y)\ f_y(x,y)\right)\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix}\right)^2$$

$$= (\Delta x\ \Delta y)\underbrace{\begin{pmatrix}\sum f_x^2(x,y) & \sum(f_x f_y)(x,y)\\ \sum(f_x f_y)(x,y) & \sum f_y^2(x,y)\end{pmatrix}}_{=:M}\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix}$$

So $\min \Delta^T M \Delta$ for $\|\Delta\| = 1$ is equivalent to maximizing the eigenvalues of M.

Harris' Measure of Corneness

$$C(x,y) = \det(M) - k(\operatorname{trace}(M))^2$$
$$= \lambda_1\lambda_2 - k(\lambda_1+\lambda_2)^2$$

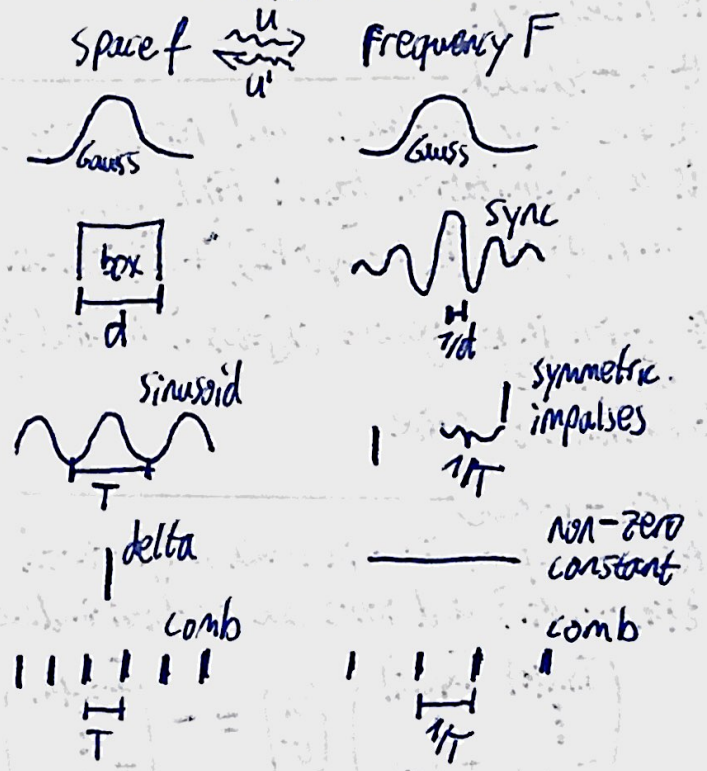
Invariant to brightness, shift, and rotation but variant to scaling.

FOURIER TRANSFORM

View a digital image as an $(X\times Y)$D vector and transform bases

$$F(g(x,y))(u,v) = \iint_{\mathbb{R}^2} g(x,y)\exp(-i2\pi(ux+vy))\,dx\,dy$$

The discrete representation in 2D has as codomain $\mathbb{C}$. In natural images, magnitude is often similar. Phase is what matters most!



Convolution theorem

$F\cdot G = \mathcal{U}(f * g)$ (cfr. filtering)

$F * G = \mathcal{U}(f\cdot g)$ (cfr. sampling)

Sampling

Can be seen as multiplying signal with comb $\leftrightsquigarrow$ repeating a spectrum in frequency domain. When spectra overlap, aliasing happens when reconstructing via box.

UNITARY TRANSFORMS

Linear transformations of a digital image f can be described through matrix-multiplication: $g = Hf$.

H unitary $\Leftrightarrow H^{-1} = H^H\ (= H^{*T})$

H orthonormal $\Leftrightarrow H$ unitary and real

Unitary transforms preserve energies:

$\|c\|^2 = c^H c = f^H A^H A f = \|f\|^2$.

PCA (Karhunen-Loève Transform)

① Let $F = (f_1\ f_2 \dots f_n)$ be a set of reference images and μ their mean. Compute their covariance matrix

$$R_{ff} = \left(\mathbb{E}[\tilde{f}_i \tilde{f}_j^H]\right)_{ij} = \frac{\tilde{F}\tilde{F}^H}{n}$$

where $\tilde{f}_i = f_i - \mu$ and $\tilde{F} = (\tilde{f}_1 \dots \tilde{f}_n)$

② Perform the eigenvalue decomposition on R_{ff}, yielding column-wise eigenvectors Φ (unitary because $R_{ff}^H R_{ff} = R_{ff} R_{ff}^H$ normal) and eigenvalues $\Lambda = \operatorname{diag}(\lambda_0, \dots, \lambda_{MN-1})$ with $\lambda_i \geq 0$ for all i because R_{ff} is symmetric non-negative definite.

③ Use $\Phi(f-\mu)$ as transform and $\Phi^T f+\mu$ as inverse transform.

PCA packs as much energy into coefficients as possible. Keeping the first $J \leq MN$ eigenvalues still forms an optimal subspace.

Fisher Linear Discriminant Analysis

Maximize between-class scatter while minimizing within-class scatter. Apply PCA first, then refine.

$$W_{opt} = \arg\max_W \left(\frac{\det|W \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T|}{\det|W \sum_{i=1}^{c} \sum_{x_k \in X_i} (x_k - \mu_i)(x_k - \mu_i)^T|} \right)$$

JPEG Encoding

8x8 block → DCT → Quantization Q → Huffman

Haar Transform

More specific than DCT within blocks, representable by filtering and downsampling using LoG pyramid.

OPTICAL FLOW

In a video, track the movement of feature points across time.

Brightness Constancy Constraint

$I(x,y,t) = I(x+\Delta x, y+\Delta y, t+\Delta t)$ for the same feature point at (x,y,t) and $(x+\Delta x, y+\Delta y, t+\Delta t)$.

Assuming small motion (< 1 pixel between frames), one approximates via Taylor

$$I(x,y,t) \approx I(x,y,t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t$$

$$\Leftrightarrow \text{Flow constraint } \partial x \quad \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t}$$

$$\Leftrightarrow \quad 0 \approx I_x u + I_y v + I_t$$

To determine u and v, another constraint is necessary:

Horn-Schnuck

Assume flow smoothness and repose problem as a minimization task: Find

$$\arg\min_{u,v} \iint_D \left((I_x u + I_y v + I_t)^2 + \alpha^2 (\|\nabla u\|^2 + \|\nabla v\|^2) \right) dx\,dy$$

⊕ motion in homogenous objects is inferred from boundaries

⊖ sensitive to noise

Lucas-Kanade

Assume equal flow in small neighborhoods $\{q_1, \dots, q_n\}$ and solve the overdetermined LSE

$$\underbrace{\begin{pmatrix} I_x(q_1) & I_y(q_1) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} u \\ v \end{pmatrix}}_{x} = - \underbrace{\begin{pmatrix} I_t(q_1) \\ \vdots \\ I_t(q_n) \end{pmatrix}}_{b}$$

Using the Linear Least Squares (LLS) method:

$$A^T A x = A^T b \Leftrightarrow x = (A^T A)^{-1} A^T b \Leftrightarrow$$

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} (I_x^2(q_i)) & \sum_{i=1}^{n} (I_x I_y(q_i)) \\ \sum_{i=1}^{n} (I_x I_y(q_i)) & \sum_{i=1}^{n} (I_y^2(q_i)) \end{pmatrix}^{-1} \begin{pmatrix} -\sum_{i=1}^{n} (I_x I_t(q_i)) \\ -\sum_{i=1}^{n} (I_y I_t(q_i)) \end{pmatrix}$$

⊕ efficient with corner feature points

⊖ breaks down in homogenous areas or on edges.

⊕ noise resistant

When movement is larger than 1 pixels between frames, one can build a Gaussian pyramid and begin at coarser levels, iteratively estimating flow and transforming the finer image until the end result is obtained at the finest level.

VIDEO COMPRESSION

Combine interlacing, image compression via DCT (abusing spatial redundancy), and temporal processing via P- and B-Frames (abusing temporal redundancy).

Bloch's Law

Stimulus Response = Intensity · Exposure Time

Videos under 10Hz appear as slide shows.

RADON TRANSFORM

Determine the values of a 2D- or 3D-function based on line integrals at discrete angles.

$$Rf(\theta, t) = \int_{L(\theta,t)} f(s)\,|ds|$$ forms a sinogram. To reconstruct f, Fourier Transform measurements for every θ, high-pass filter, and IFT to obtain $q_\theta(t)$. Now, back-project

$$f_{bp}(x,y) = \int q_\theta(x\cos\theta + y\sin\theta)\,d\theta$$

GRAPHICS PIPELINE

The process of going from geometry, materials, and lighting definitions to pixels nowadays encompasses

Modeling Transform → Viewing Transform → Primitive Processing → 3D-Clipping → Screen-Space Projection → Scan Conversion → Lighting, Shading, Texturing → Occlusion Handling → Display

Programmatic control

CPU → vertex processing → rasterization → fragment processing → D

attr. uniforms | Varyi. | interpolation | Varyi. uniforms | color

AFFINE TRANSFORMATIONS

translate $\begin{pmatrix}x\\y\end{pmatrix}+\begin{pmatrix}t_x\\t_y\end{pmatrix}$ 

scale $\begin{pmatrix}s_x & 0\\ 0 & s_y\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}$ 

rotate $\begin{pmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}$ 

Homogeneous Coordinates

Raise dimensionality by one to make all affine transformations linear.

$\begin{pmatrix}x\\y\end{pmatrix} \overset{\text{hom.}}{\rightsquigarrow} \begin{pmatrix}xw\\yw\\w\end{pmatrix} \overset{\text{cart.}}{\rightsquigarrow}$ with $w\in\mathbb{R}\setminus\{0\}$

Now, translate, rotate, and scale through left-multiplication with

translate $\begin{pmatrix}1&0&t_x\\0&1&t_y\\0&0&1\end{pmatrix}$ scale $\begin{pmatrix}s_x&0&0\\0&s_y&0\\0&0&1\end{pmatrix}$ 2D-rotate $\begin{pmatrix}\cos\theta&-\sin\theta&0\\\sin\theta&\cos\theta&0\\0&0&1\end{pmatrix}$

3D-rotate(x) $\begin{pmatrix}1&0&0&0\\0&\cos\theta&-\sin\theta&0\\0&\sin\theta&\cos\theta&0\\0&0&0&1\end{pmatrix}$ 3D-rotate(y) $\begin{pmatrix}\cos\theta&0&\sin\theta&0\\0&1&0&0\\-\sin\theta&0&\cos\theta&0\\0&0&0&1\end{pmatrix}$ 3D-rotate(z) $\begin{pmatrix}\cos\theta&-\sin\theta&0&0\\\sin\theta&\cos\theta&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}$

Change coordinate systems with

$$p' = \begin{pmatrix} | & | & | & | \\ r_1 & r_2 & r_3 & t \\ | & | & | & | \\ 0 & 0 & 0 & 1 \end{pmatrix} p$$

Transform normals with

$$p' = Mp \Rightarrow n' = (M^{-1})^T n$$

Quaternions

perform rotations and translations efficiently in a 4D associative normed division algebra.

$\begin{pmatrix}x\\y\\z\end{pmatrix}$ quat. / cart. $\quad q + xi + yj + kz$

·	i	j	k
i	-1	k	-j
j	-k	-1	i
k	j	-i	-1

vector notation $q + \begin{pmatrix}x\\y\\z\end{pmatrix}$

$(s_1+\vec{v}_1)(s_2+\vec{v}_2)$

$= s_1 s_2 - \vec{v}_1\cdot\vec{v}_2 + s_1\vec{v}_2 + s_2\vec{v}_1 + \vec{v}_1\times\vec{v}_2$,

$\overline{(s+\vec{v})} = s-\vec{v}$, $z\bar{z} = \|z\|_2^2$

$z^{-1} = \frac{\bar{z}}{\|z\|^2}$, $1 = z^{-1}z = zz^{-1}$

$\begin{bmatrix} y_1z_2 - z_1y_2 \\ z_1x_2 - x_1z_2 \\ x_1y_2 - y_1x_2 \end{bmatrix}$

Now, rotate by θ around unit axis u by

$p' = qpq^{-1}$ with $q = \cos(\frac{\theta}{2}) + \sin(\frac{\theta}{2})u$

In comparison to Euler angles quaternions do not suffer from gimbal lock.

SHADING & LIGHTING

Flux $\Phi(A)$ $[\frac{J}{s}=W]$ energy passing through space A per time

Radiosity $B(x) = \frac{d\Phi(A)}{dA(x)}$ $[\frac{W}{m^2}]$ flux per unit leaving surface

Irradiance $E(x) = \frac{d\Phi(A)}{dA(x)}$ $[\frac{W}{m^2}]$ flux per unit arriving on surface

Rad. Intensity $I(\vec{w}) = \frac{d\Phi}{d\vec{w}}$ $[\frac{W}{sr}]$ flux per solid angle emanating from point source

Radiance $L(x,\vec{w}) = \frac{d^2\Phi(A)}{\cos\theta\, dA(x)\, d\vec{w}}$ $[\frac{W}{m^2 sr}]$

flux per unit solid angle in ray direction and projected area perpendicular to the ray

BRDF

$$f_r(x,\vec{w_i},\vec{w_r}) = \frac{dL_r(x,\vec{w_r})}{dE_i(x,\vec{w_i})} = \frac{1}{L_i(x,\vec{w_i})\cos\theta_i}\cdot\frac{dL_r(x,\vec{w_r})}{d\vec{w_i}}$$

The relation between incident radiance and differential reflected radiance forms the reflection equation for local illumination:

$L_r(x,\vec{w_r}) = \int_{H^2} f_r(x,\vec{w_i},\vec{w_r}) L_i(x,\vec{w_i}) \cos\theta_i d\vec{w_i}$

physically-based BRDFs satisfy $f_r \geq 0$, $f_r(\vec{w_i},\vec{w_r}) = f_r(\vec{w_r},\vec{w_i})$ (Helmholtz), energy preservation

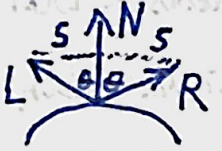
Phong Illumination Model

$$I = I_a k_a + I_p(k_d(N\cdot L) + k_s(R\cdot V)^n)$$

ambient — diffuse — specular

with material parameters k_a, k_d, k_s, n
light intensities I_a, I_p
surface normal N, light ray L,
view ray V (all normalized)

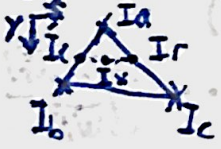
and $R = N\cos\theta + S$
$= 2N\cos\theta - L$
$= 2N(N\cdot L) - L$



Shading

Flat: one color per primitive
-or-
① Calculate face normals.
② Calculate vertex normals by averaging.

Gouraud: Linearly interpolate vertex intensities
③ Evaluate illumination model per vertex.
④ Interpolate color bilinearly on scan line



$I_e = \text{lerp}\left(I_a, I_b, \frac{y_x - y_a}{y_b - y_a}\right)$

$I_r = \text{lerp}\left(I_a, I_c, \frac{y_x - y_a}{y_c - y_a}\right)$

$I_x = \text{lerp}\left(I_e, I_r, \frac{x_x - x_l}{x_r - x_l}\right)$

Phong: Linearly interpolate normals
③ Interpolate normal barycentric

$n_x = \frac{\cdot\cdot}{\cdot\cdot\cdot} n_a + \frac{\cdot}{\cdot\cdot\cdot} n_b + \frac{\cdot\cdot}{\cdot\cdot\cdot} n_c$

④ Evaluate illumination model per fragment.

Transparency

Alpha blending is the linear interpolation of color, front-to-back:

$$I = I_1\alpha_1 + I_2(1-\alpha_2)$$

So, render back-to-front, beginning with the opaque object.
Overlap requires multi-pass depth peeling.

GEOMETRY & TEXTURE

Parametric, subdivision, point set, implicit, polygonal (vertices, edges, faces) representations of geometry make sense.

Triangle storage

List: vertex, vertex, vertex ⇒ redundancy
indexed face set: id, id, id; id, vertex

(u,v)-mapping describes texture-to-vertex relationship. Parametrization matters!
e.g. sphere $\begin{bmatrix} u \\ v \end{bmatrix} \mapsto \begin{bmatrix} \sin(u)\sin(v) \\ \cos(v) \\ \cos(u)\sin(v) \end{bmatrix}$

Mipmapping

Generate a set of progressively lower resolution versions for each texture using a Gaussian pyramid, and index based on projected triangle area. Memory overhead is $\sum_{n=1}^{\infty} \frac{1}{4^n} = \frac{1}{3}$ of the full-res texture size.

Bump Mapping

With Phong shading, displace normal vectors.
Illusion of geometry but incorrect silhouettes.

CONTROLLED CURVES

Bézier

Bernstein polynomial $B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$
evaluated at all t or de Casteljau's algorithm (repeated lerps).
⊕ approximating curve within bounds of control point polygon
⊖ global support

B-Splines

⊕ local support with fixed degree
⊖ No C^2-continuity

Natural Splines

⊕ piecewise degree 3 with C^2-continuity
⊖ global support

RIGID-BODY DYNAMICS

center of mass $c(t) = \frac{\int \rho x\,dx}{\int \rho\,dx}$

translation $x(t)$ with $\dot{x}(t) = v(t)$, $\dot{v}(t) = a(t)$

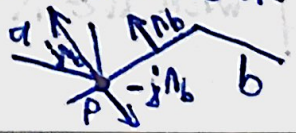
rotation quaternion $q(t)$ with $\dot{q}(t) = \frac{1}{2}w(t)q(t)$

linear momentum $P(t)$ with $P'(t) = F(t) = ma(t)$

angular momentum $L(t)$ with $L'(t) = S(t) = (x(t) - c(t)) \times$

Collision Detection

bodies a and b collide in $p \Rightarrow v_{rel} < 0$
where $v_{rel} = n_b(\dot{p}_a - \dot{p}_b)$



impulse response j

Collision Response

$$j = \frac{-(1+\varepsilon)v_{rel}}{\frac{1}{m_a} + \frac{1}{m_b} + n\left(\left(I_a^{-1}(r_a \times n)\right) \times r_a + \left(I_b^{-1}(r_b \times n)\right) \times r_b\right)}$$

with bounciness ε (1 = elastic)
masses m_a, m_b
center-of-mass-to-p vectors r_a, r_b
Form impulse along normal $J = jn$
which changes P by $\Delta P = J$ and L by $\Delta L = (p - c(t)) \times J$, making $v_{rel}^{+} = -\varepsilon v_{rel}^{-}$

Forward Euler

$x(t+\Delta t) = x(t) + \Delta t\, \dot{x}(t)$
⊕ predict fully based on current state
⊖ instable

Symplectic Euler

$x(t+\Delta t) = x(t) + \Delta t\, \dot{x}(t+\Delta t)$
$\dot{x}(t+\Delta t) = \dot{x}(t) + \Delta t\, \ddot{x}(t)$
⊕ mostly energy conserving, forward-solvable
⊖ not generally applicable for >2nd order ODEs

Backward Euler

Predict value through future velocity using implicit configurations
⊕ stable
⊖ backward-solvable, computationally expensive, numerical damping

SCAN CONVERSION

Bresenham Line

Given a line equation $f(x,y) = ax + by + c \doteq 0$ and a previous pixel (x_p, y_p), select based on $f(x_p + 1, y_p + \frac{1}{2})$. If >0, select (x_p+1, y_p+1), otherwise (x_p+1, y_p).

Polygon Scan Conversion

① Calculate all intersections with the scan line.
② Sort by ascending x-coordinates.
③ Fill all spans if their parity is odd.

VISIBILITY & SHADOWS

Painter's Algorithm

Render polygons from furthest to nearest.
Problem: cyclic overlaps, intersections

z-Buffering

Per-pixel visibility decision based on its z-index.
Problem: limited codomain, non-linear resolution

Planar Shadows

Draw the projection of an object onto the flat ground.
Problem: flat, no self-shadowing, no shadows on other objects

Projective Texture Shadows

Separate obstacle from receiver, compute a blur image of the obstacle from the light source, and use it as a projective texture.
Problem: manual specification required, no self-shadows.

Shadow Maps

① In pre-production, compute a 2D-depth image from every light source.
② At runtime, cast one ray per pixel into the scene, project point onto the light planes and compare depths to determine shadow status
Problem: self-shadowing from numerical imprecision (fix: additive bias), aliasing from undersampled shadow map (fix: supersampling or filtering → soft edges)

Shadow Volumes

① In pre-production, compute the geometry of the shadow volume
② At runtime, track the intersections of camera rays with shadow volumes, incrementing on entry and decrementing on exit. Determine shadow status through counter $\neq 0$.
Problem: plentiful new geometry, expensive due to ray casts

RAY TRACING

From the eye point, shoot a primary ray through every pixel. On hit, send shadow rays to all light sources and secondary rays depending on desired illumination model. Terminate when satisfied.

Anti-Aliasing

Supersampling sends multiple, minimally perturbated rays through each pixel and averages the result.

Ray-Surface Intersections

ray equation $r(t) = o + td$

Sphere test: $\|o + td - c\|^2 - r^2 = 0$

triangle intersect plane $t = -\frac{(o - p_1) \cdot n}{d \cdot n}$

where $n = (p_2 - p_1) \times (p_3 - p_1)$

triangle point-on-plane inside?
→ barycentric coordinates in $[0,1]$ and sum to 1.

Uniform Grids

Accelerate spatial searches through equi-sized bucket volumes with mappings to contained polygons.

⊕ Easy to compute, fast to implement
⊖ Doesn't scale with scene complexity

Space-Partitioning Trees

Optimize discretization by maximizing block size and minimizing number of contained primitives.