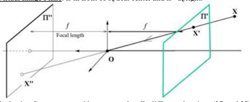
Allen Zou (A15016042)

Introduction and Overview

Four Rs of Computer Vision
Reprojection: Rendering scene from different view, illumination, surface properties, etc.
Reconstruction: Multiple view geometry, structure from motion, shape from X (X = shading, texture, contour).
Registration: Tracking, alignment, optical flow, etc.
Recognition: recognizing objects, scenes, events, etc.

Geometric Image Formation

Two sensor types:
CCD: high sensitivity and power, can't individually addressed, blooming.
CMOS: simple to fabricate, lower sensitivity and power, can be individually addressed.
Virtual Image Plane: is in front of optical center and is "upright."



Projective Geometry: provides a way to handle different situations (3D and 2D) in a unified way.
Homogeneous coordinates allow for points at infinity to be treated like other points. In projective plane, parallel lines meet at point infinity. 2D vanishing point = perspective projection of 3D point at infinity.

$$\infty = \lim_{x \to 0^+} \frac{1}{x} \qquad \mathbf{X}_\infty = (x, y, 0)^\top$$
$$0 = \lim_{x \to +\infty} \frac{1}{x} \qquad \mathbf{X}_\infty = (X, Y, Z, 0)^\top$$

Distance of line from origin: c/sqrt(a^2 + b^2) = c/||n||
2D Points and Lines
Joining a line

$$\begin{bmatrix}\mathbf{x}_1^\top \\ \mathbf{x}_2^\top\end{bmatrix} \boldsymbol{\ell} = \mathbf{0} \qquad \boldsymbol{\ell} = \mathbf{x}_1 \times \mathbf{x}_2$$
$$\boldsymbol{\ell} = [\mathbf{x}_1]_\times \mathbf{x}_2 = (\mathbf{x}_1^\top [\mathbf{x}_2]_\times)^\top$$
$$[\boldsymbol{\ell}]^\perp = \begin{bmatrix}\mathbf{x}_1^\top \\ \mathbf{x}_2^\top\end{bmatrix} \quad [\boldsymbol{\ell}]_\times = \mathbf{x}_1 \mathbf{x}_2^\top - \mathbf{x}_2 \mathbf{x}_1^\top$$

Intersection

$$\begin{bmatrix}\boldsymbol{\ell}_1^\top \\ \boldsymbol{\ell}_2^\top\end{bmatrix} \mathbf{x} = \mathbf{0} \qquad \mathbf{x} = \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2$$
$$\mathbf{x} = [\boldsymbol{\ell}_1]_\times \boldsymbol{\ell}_2 = (\boldsymbol{\ell}_1^\top [\boldsymbol{\ell}_2]_\times)^\top$$
$$[\mathbf{x}]^\perp = \begin{bmatrix}\boldsymbol{\ell}_1^\top \\ \boldsymbol{\ell}_2^\top\end{bmatrix} \quad [\mathbf{x}]_\times = \boldsymbol{\ell}_1 \boldsymbol{\ell}_2^\top - \boldsymbol{\ell}_2 \boldsymbol{\ell}_1^\top$$

Skew Matric (rank 2)

$$\mathbf{a} = (a_1, a_2, a_3)^\top$$
$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

Camera: Mathematical expression relating points in 3D to points in image for different types of physical cameras or imaging situations.
Information About Coordinate Systems

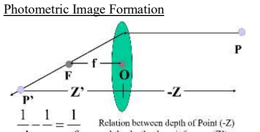
$$R = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$
(Z, Y, X)

$$\tilde{0}_{\text{cam}} = R\tilde{C} + t \qquad \alpha_x = fm_x \qquad \mathbf{x} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}] \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{X}$$
$$-R\tilde{C} = t \qquad \alpha_y = fm_y \qquad \mathbf{x} = \mathtt{K}[\mathtt{R} \mid \mathbf{t}]\mathbf{X}$$
$$\tilde{C} = -R^\top t \qquad x_0 = m_x p_x \qquad \mathbf{x} = \mathtt{P}\mathbf{X}$$
$$y_0 = m_y p_y$$

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_X \\ r_{21} & r_{22} & r_{23} & t_Y \\ r_{31} & r_{32} & r_{33} & t_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix}$$

Deviation from lens (geometric and chromatic abberations)
Spherical Aberration: rays parallel to axis don't converge. Outer portions of the lens yield smaller focal lengths.
Astigmatism: rays that propagate in two perpendicular planes have different focus – vertical and horizontal lines will be in sharp focus at two distances.
Distortion: focal length different for diff. angles of inclination.
Coma: points off the axis are depicted as a comet-shaped blob.
Chromatic: rays of different wavelengths focused in different planes (can't be removed completely).
Other camera models:
Generalized camera: maps points lying on rays to points on image plane. Omnicam (hemispherical) and light probe (spherical).

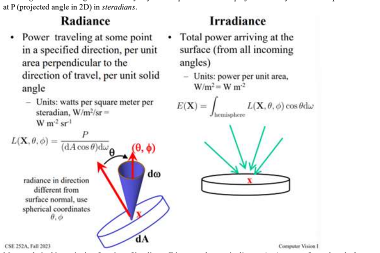
Photometric Image Formation



A price: Whereas the image of P is in focus, the image of Q is not.

$$\frac{1}{z'} - \frac{1}{z} = \frac{1}{f}$$ Relation between depth of Point (-Z) and the depth where it focusses (Z')

Thin Lens: Aperture
Smaller means less blur and pinhole means no blur.
Measuring Angle
Solid angle: The solid angle subtended by object from point P is area of projection of object onto unit sphere centered at P (projected angle in 2D) in *steradians*.



Measured pixel intensity is a function of irradiance E integrated over pixel's area (x,y), range of wavelengths λ, some period of time t. R is the camera response function.

$$I = R\left(\int_t \int_\lambda \int_x \int_y E(x, y, \lambda, t) s(x, y) q(\lambda)\, dx\, dy\, d\lambda\, dt\right)$$
Spatial response s(x,y) Spectral response q(λ)

For a camera with a thin lens, E(x) = k_L * L.
Combined with linear sensor model, I = k_s * k_L * L.
E(x) = image irradiance at point x.
L = radiance coming from a scene point projecting to image point x.
k_L = proportionality constant (may depend on lens and x).
k_s = linear sensor model.
Measured pixel intensity is proportional to radiance.
Bi-directional Refelctance Distribution Function:
ρ(θ_{in}, ϕ_{in}; θ_{out}, ϕ_{out}) – ratio of radiance to irradiance.
General: No assumption of symmetry or directional characteristics - wide range of reflection behaviors.
Isotropic: BRDF is symmetric about surface normal for same incident and emitting directions.
Anisotropic: Has directional variations in reflection properties – with preferential reflectance directions.
Reflection
Specular (like mirror): ω_o = 2*(ω_i * N) * N – ω_i
Diffuse: incident light per unit area∝cos(θ) = $n^T s$
Glossy: like mirror, but some light reflected slightly off ideal reflection direction.
Ambient: light bounced around scene (not fully dark).
Ways to measure BRDF
Gonioreflectometers: 3/4 degrees of freedom (isotropic/anisotropic) for light source, detector, sample.
Marschner's Image-Based BRDF: For uniform BRDF, capture 2D slice corresponding to variations in normals.
Ward's BRDF Setup: Collect refelcted light with hemispherical (should be ellipsoidal) mirror - each image captures light at all exitant angles.
Light: Internally generated power radiated per unit area on surface (AKA radiant emittance) – how bright/what color are the objects. Simiar to irradiance with units of W/m^2.
Point source *Distant source* Don't allow θ > 90

$$\rho_d(x)\left(\frac{N(x)^T S(x)}{r(x)^2}\right) \quad \rho_d(x)\left(N(x)^T S(x)\right)$$
$$\cos^+(\theta) = \max(0, \hat{\mathbf{n}}^\top \hat{\mathbf{s}})$$

Area source: leads to soft penumbra shadows.
Lambertian (Diffuse) Reflection

$$e(x, y) = a(x, y) s_0 \cos\theta$$
$$e(x, y) = a(x, y) s_0 \hat{\mathbf{n}}(x, y)^\top \hat{\mathbf{s}}(x, y), \text{ where } \cos\theta = \hat{\mathbf{n}}(x, y)^\top \hat{\mathbf{s}}(x, y)$$
$$e(x, y) = a(x, y) s_0 \max(0, \hat{\mathbf{n}}(x, y)^\top \hat{\mathbf{s}}(x, y))$$
$$e(x, y) = a(x, y) \max(0, \hat{\mathbf{n}}(x, y)^\top \mathbf{s}(x, y)), \text{ where } \mathbf{s}(x, y) = s_0 \hat{\mathbf{s}}(x, y)$$

Regarding color, surface color and light color are RGB. Must compute RGB values of reflected color separately.
Blinn-Phong (Ambient + Specular too) Reflection

$$e(x, y) = s_{a,0} k_a(x, y) + s_0 (k_d(x, y) f_d(x, y) + k_s(x, y) f_s(x, y))$$
where
$$f_d(x, y) = \max(0, \hat{\mathbf{n}}(x, y)^\top \hat{\mathbf{s}}(x, y))$$
$$f_s(x, y) = \max(0, \hat{\mathbf{n}}(x, y)^\top \hat{\mathbf{h}}(x, y))^{\alpha(x, y)}$$
where
$$\hat{\mathbf{h}}(x, y) = \frac{1}{\|\mathbf{h}(x, y)\|} \mathbf{h}(x, y) \text{ and } \mathbf{h}(x, y) = \hat{\mathbf{s}}(x, y) + \hat{\mathbf{v}}(x, y)$$

ka(x,y), kd(x,y), ks(x,y), is the ambient, diffuse, specular value of the surface facet imaged by pixel.
α(x,y) is the shininess of the surface facet imaged by pixel.
n_hat(x,y) is the unit normal of the surface facet imaged by pixel.
s_a,0 is the ambient light intensity while s is the intensity of the light source.
s_hat(x,y) is the unit direction to the light source from the surface facet imaged by pixel.
v_hat(x,y) is the unit direction to the camera from the surface facet imaged by pixel.

Photometric Stereo

Shape-From-X Methods
Shape-from-shading: One image – requires knowledge of light source direction and BRDF everywhere.
Photometric Stereo: One viewpt, multiple images/lighting.
General BRDF, Known Lighting
Gradient Space (p, q)

$$\text{Surface: } \mathbf{s}(x, y) = (x, y, f(x, y))$$
$$\text{Tangent vectors: } \frac{\partial s(x, y)}{\partial x} = \left(1, 0, \frac{\partial f}{\partial x}\right) \quad \frac{\partial s(x, y)}{\partial y} = \left(0, 1, \frac{\partial f}{\partial y}\right)$$
Normal vector
$$\mathbf{n} = \boxed{\frac{\partial s}{\partial x} \times \frac{\partial s}{\partial y}} = \left(-\frac{\partial f}{\partial x}, -\frac{\partial f}{\partial y}, 1\right)$$

Reflectance Map: Assume BRDF same all points and light direction constant. Image irradiance is function of only direction of normal. We have E(p, q) in gradient space.

$$E = L\rho \frac{1 + pp_s + qq_s}{\sqrt{1 + p^2 + q^2}\sqrt{1 + p_s^2 + q_s^2}}$$

Three Source Photometric Stereo
Construct 3 reflectance R(p,q) maps for 3 light source directions and BRDF. Aquire 3 E(x,y) from 3 images with known light source directions. For each (x,y) pixel location, find (p,q) as intersection of 3 curves R(p,q)=E(x,y). (p,q) is the surface normal at pixel (x,y).
Recovering the surface f(x,y)
From estimate n=(n_x,n_y,n_z), p=-n_x/n_z, q= n_y/n_z, set f(0,0)=0 and integrate p=df/dx along row (x,0) to get f(x,0). Then integrate q=df/dy along columns starting with first row.
Note: p and q estimated independently. Noise means integrating along a curve might not return to original point.
Horn's Method
Formulate estimation of surface height z(x,y) from gradient field by minimizing cost functional:

$$\iint_{\text{Image}} (z_x - p)^2 + (z_y - q)^2\, dx\, dy$$

where z_x and z_y are partial derivatives of best fit surface. Solved using calculus of variations – iterative updating. z(x,y) can be discrete or represented in terms of basis functions. Integrability is naturally satisfied.
Lambertial Surface, Known Lighting

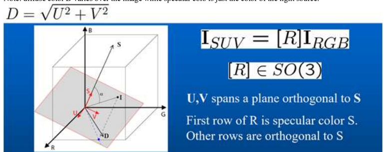
$$e_i(x, y) = a(x, y) \hat{\mathbf{n}}(x, y)^\top s_{i,0} \hat{\mathbf{s}}_i, \text{ solve for } a(x, y) \text{ and } \hat{\mathbf{n}}(x, y)$$
$$e_i(x, y) = \mathbf{b}(x, y)^\top \mathbf{s}_i, \text{ where } \mathbf{b}(x, y) = a(x, y) \hat{\mathbf{n}}(x, y) \text{ and } \mathbf{s}_i = s_{i,0} \hat{\mathbf{s}}_i$$
$$e_i(x, y) = \mathbf{s}_i^\top \mathbf{b}(x, y)$$
$$\begin{bmatrix} e_1(x, y) \\ e_2(x, y) \\ \vdots \\ e_n(x, y) \end{bmatrix} = \begin{bmatrix} \mathbf{s}_1^\top \\ \mathbf{s}_2^\top \\ \vdots \\ \mathbf{s}_n^\top \end{bmatrix} \mathbf{b}(x, y) \qquad n \geq 3 \text{ light sources and associated i}$$
$$\mathbf{e}(x, y) = \mathtt{S}\mathbf{b}(x, y), \text{ solve for } \mathbf{b}(x, y)$$
$$\mathtt{S}^\top \mathbf{e}(x, y) = \mathtt{S}^\top \mathtt{S} \mathbf{b}(x, y)$$
$$(\mathtt{S}^\top \mathtt{S})^{-1} \mathtt{S}^\top \mathbf{e}(x, y) = \mathbf{b}(x, y)$$
$$\mathtt{S}^+ \mathbf{e}(x, y) = \mathbf{b}(x, y), \text{ where } \mathtt{S}^+ = (\mathtt{S}^\top \mathtt{S})^{-1} \mathtt{S}^\top$$
Same for all pixel coordinates
$$\mathbf{b}(x, y) = a(x, y) \hat{\mathbf{n}}(x, y), \text{ where } a(x, y) = \|\mathbf{b}(x, y)\| \text{ and } \hat{\mathbf{n}}(x, y) = \frac{\mathbf{b}(x, y)}{\|\mathbf{b}(x, y)\|}$$

Lambertian BRDF, Unknown Lighting: uncalibrated stereo
For calibrated stereo, we estimate Nx3 matrix B of surface normals scaled by albedo using lighting. For uncalibrated stereo: only images, no lighting info. Without shadowing, all images lie in 3D subspace of the N-pixel image space spanned by columns of an Nx3 matrix B*. From 3+ images, use SVD to estimate B*. The Nx3 matrix B of surface normals scaled by albedo differs from B* by a 3x3 linear transformation B=AB*/ After enforcing integrability, one can only estimate shape and albedo (B) up to a Generalized Bas Relief (GBR) tranformation, which has 3 parameters (depth scaling, tilt).
SUV Color Space: Photometric Stereo of Glossy Objects

$$I_k = (D_k f_d + S_k f_s(\boldsymbol{\theta}))\, \hat{\mathbf{n}} \cdot \hat{\mathbf{l}} \quad \boldsymbol{\theta} = (\theta_i, \phi_i, \theta_r, \phi_r)$$
Color channel k
$$L(\lambda) = \text{Spectral Power Distribution of light source}$$
$$C_k(\lambda) = \text{Camera Sensitivity}$$
$$S_k = \int C_k(\lambda) L(\lambda)\, d\lambda. \qquad \text{Specular Color}$$
$$D_k = \int C_k(\lambda) L(\lambda) g_d(\lambda)\, d\lambda \qquad \text{Diffuse Color}$$
$$\begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix} = (f_d \hat{\mathbf{n}} \cdot \hat{l}) \begin{bmatrix} D_r \\ D_g \\ D_b \end{bmatrix} + (f_s(\theta) \hat{\mathbf{n}} \cdot \hat{l}) \begin{bmatrix} S_r \\ S_g \\ S_b \end{bmatrix} \qquad \mathbf{I} = (f_d \hat{n} \cdot \hat{l})\mathbf{D} + (f_s(\theta) \hat{n} \cdot \hat{l})\mathbf{S}$$

Where f_d and f_s are diffuse and specular BRDF. Image color lies in span of diffuse color D and specular color S.
Note: diffuse color D varies over the image while specular colo is just the color of the light source.

$$D = \sqrt{U^2 + V^2}$$



Given color of light source $\mathbf{c} = (R_s, G_s, B_s)^\top$, calculate $\mathtt{R} \in SO(3)$ such that $(1, 0, 0)^\top = \mathtt{R}\hat{\mathbf{c}}$

$$\hat{\mathbf{c}} = \frac{\mathbf{c}}{\|\mathbf{c}\|}$$
$$\mathtt{R} = \begin{bmatrix} \hat{\mathbf{c}}^\top \\ [\hat{\mathbf{c}}]^{\perp} \end{bmatrix}, \text{ where } [\hat{\mathbf{c}}]^\perp \hat{\mathbf{c}} = \mathbf{0} \text{ (i.e., } [\hat{\mathbf{c}}]^\perp \text{ is left null space of } \hat{\mathbf{c}}\text{)}$$
if det(R) = −1, then negate last row of R

Properties of SUV: Data-dependent, rotational transformation. S channel encodes entire specular component and unknown amount of diffuse component. Shading info is preserved in U & V channels.
Multi-channel Photometric Stereo

$\mathbf{J} = [I_U \;\; I_V]^\top$
$\mathbf{J}^k$: 2-channel color vector under the k^{th} light source.
$\hat{\mathbf{l}}^k$: The k^{th} three light source directions.
$\boldsymbol{\rho}$: 2-channel UV albedo.

$$\mathbf{J}^k = [I_U^k, I_V^k]^\top = (\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}^k)\boldsymbol{\rho},$$

$$\text{Shading vector} \quad = \mathbf{F} \quad = \left[f^1, f^2, f^3\right]^\top \quad = [\hat{\mathbf{l}}^1 \; \hat{\mathbf{l}}^2 \; \hat{\mathbf{l}}^3]^\top \hat{\mathbf{n}}$$
$$\text{Intensity matrix} \quad = [J] \quad = \begin{bmatrix} J_1^1 & J_2^1 \\ J_1^2 & J_2^2 \\ J_1^3 & J_2^3 \end{bmatrix} = \begin{bmatrix} f^1 \rho_U & f^1 \rho_V \\ f^2 \rho_U & f^2 \rho_V \\ f^3 \rho_U & f^3 \rho_V \end{bmatrix} = \mathbf{F}\boldsymbol{\rho}^\top.$$

The least squares estimate of the shading vector $\mathbf{F}$ is the principal eigenvector of $[J][J]^\top$. Once the shading vector is known, the surface normal is found by solving the matrix equation $\mathbf{F} = [\hat{\mathbf{l}}^1 \; \hat{\mathbf{l}}^2 \; \hat{\mathbf{l}}^3]^\top \hat{\mathbf{n}}$.

Image Filtering

Process of linear filters: form new image whose pixels are weighted sum of original pixel values, using same weights each point. Properties: output is linear function of input. Output is shift-invariant function of input.

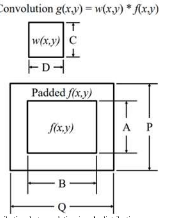
"Shape" of correlation/convolution

Convolution $g(x,y) = w(x,y) * f(x,y)$

Full
- w(x,y) and f(x,y) have at least 1 pixel overlap
- P = A + 2(C − 1)
- Q = B + 2(D − 1)
- Output g(x,y)
  - Width is B + D − 1, height is A + C − 1

Same
- P = A + C − 1
- Q = B + D − 1
- Output g(x,y)
  - Width is B, height is A

Valid
- w(x,y) must be completely inside f(x,y)
- P = A
- Q = B
- Output g(x,y)
  - Width is B − D + 1, height is A − C + 1



Convolution is commutative, associative, and distributive, but correlation is only distributive.
Continuous convolution: commutative, associative, linear, and differentiale rule.

$$f * g(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x - u, y - v) g(u, v)\, du\, dv$$

Gaussian Kernel: Minimum size is ceil(6σ). Round up if the value is even.

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2 + y^2)}{2\sigma^2}}$$

Image Processing in the Frequency Domain
Image spacial domain f(x,y) → Fourier transform → Image frequency domain F(u,v) → Frequency domain processing → Image frequency domain G(u,v) → Inverse Fourier tranform → Image spatial domain g(x,y).
Fourier Transform: Consider time domain signal expressed as weighted sum of sinusoids cos(ut+Φ), characterized by its phase Φ and frequency u. Fourier transform of signal is a function giving weights and phase as a function of frequency u (from real to complex values).
Discrete Fourier Transform

$$F[u, v] \equiv \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} I[x, y] e^{\frac{-2\pi i}{N}(xu + yv)}$$
$$I[x, y] \equiv \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F[u, v] e^{\frac{+2\pi i}{N}(ux + vy)}$$

Fourier Basis Element

$$e^{-i2\pi(ux + vy)}$$

Above equation transforms the sum of orthogonal basis functions.
Vector (u,v): Magnitude gives frequency and direction gives orientation.
Phase and Magnitude: $e^{i\theta}$ = cos(θ) + i*sin(θ)
Fourier transform of a real function is complex, which is difficult to plot and visualize. Instead, think of phase and magnitude. Phase and magnitude are the phase and magnitude of complex tranform respectively. All natural images have about the same magnitude transform. Phase seems to matter, but magnitude largely doesn't.
The Fourier Transform and Convolution Theorem
If H and G are images, and F(.) represents Fourier transform, then F(H*G) = F(H)F(G). Can think of properties of convolution as how it modifies frequencies of image. Ex: convolving image H by G attenuates frequences where G has low power, and amplifies those which have high power.
Various Fourier Transform Pairs
Scaling a function down in spatial domain scales up Fourier Transform in the frequency domain. High-frequency components in frequency domain correspond to small details in spatial domain (can use this to fine-tune details). The Fourier Transform of Gaussian is a Gaussian.
Other Types of Noise
Impulsive: randomly pick a pixel and set random value.
Quantization effects: map from continuous to discrete.
Unanticipated image structures: artifacts, anomalies, etc.

Edge Detection and Corner Detection

First/second derivative: [-1/2 0 1/2] and [1 -2 1]
Three major issues with computing gradients in 2D: The gradient magnitude at different scales is different. Gradient magnitude is large along thick trail. How to link relavent points up into curves?

Calibrated Stereo (Part 1)

Stereo Vision Outline: calibrate stereo cameras, aquire stereo images, epipolar rectify stereo images, establish correspondences, and estimate depth.
Compute R and t of second camera relative to the first one

$$\mathbf{x}_1 = \mathtt{K}_1[\mathtt{I} \mid \mathbf{0}]\begin{bmatrix} \mathtt{R}_1 & \mathbf{t}_1 \\ \mathbf{0}^\top & 1 \end{bmatrix}\mathbf{X} \qquad \mathbf{x}_2 = \mathtt{K}_2[\mathtt{I} \mid \mathbf{0}]\begin{bmatrix} \mathtt{R}_2 & \mathbf{t}_2 \\ \mathbf{0}^\top & 1 \end{bmatrix}\mathbf{X}$$
$$\mathbf{x}_1 = \mathtt{K}_1[\mathtt{I} \mid \mathbf{0}]\mathbf{X}_{\text{cam},1} \qquad \mathbf{x}_2 = \mathtt{K}_2[\mathtt{I} \mid \mathbf{0}]\begin{bmatrix} \mathtt{R}_2 & \mathbf{t}_2 \\ \mathbf{0}^\top & 1 \end{bmatrix}\begin{bmatrix} \mathtt{R}_1 & \mathbf{t}_1 \\ \mathbf{0}^\top & 1 \end{bmatrix}^{-1}\mathbf{X}_{\text{cam},1}$$
$$\text{where } \mathbf{X}_{\text{cam},1} = \begin{bmatrix} \mathtt{R}_1 & \mathbf{t}_1 \\ \mathbf{0}^\top & 1 \end{bmatrix}\mathbf{X} \qquad \mathbf{x}_2 = \mathtt{K}_2[\mathtt{I} \mid \mathbf{0}]\begin{bmatrix} \mathtt{R}_2 & \mathbf{t}_2 \\ \mathbf{0}^\top & 1 \end{bmatrix}\begin{bmatrix} \mathtt{R}_1^\top & -\mathtt{R}_1^\top\mathbf{t}_1 \\ \mathbf{0}^\top & 1 \end{bmatrix}\mathbf{X}_{\text{cam},1}$$
$$\begin{bmatrix} \mathtt{R}_1 & \mathbf{t}_1 \\ \mathbf{0}^\top & 1 \end{bmatrix}^{-1}\mathbf{X}_{\text{cam},1} = \mathbf{X} \qquad \mathbf{x}_2 = \mathtt{K}_2[\mathtt{I} \mid \mathbf{0}]\begin{bmatrix} \mathtt{R}_2\mathtt{R}_1^\top & \mathbf{t}_2 - \mathtt{R}_2\mathtt{R}_1^\top\mathbf{t}_1 \\ \mathbf{0}^\top & 1 \end{bmatrix}\mathbf{X}_{\text{cam},1}$$
$$\text{where } \boxed{\begin{matrix}\mathtt{R} = \mathtt{R}_2\mathtt{R}_1^\top \text{ and} \\ \mathbf{t} = \mathbf{t}_2 - \mathtt{R}_2\mathtt{R}_1^\top\mathbf{t}_1\end{matrix}} \qquad \mathbf{x}_2 = \mathtt{K}_2[\mathtt{I} \mid \mathbf{0}]\begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}\mathbf{X}_{\text{cam},1}$$

Image Points
Image points in pixel coordinates
$$\mathbf{x} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}]\begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}\mathbf{X}$$
$$\mathbf{x} = \mathtt{K}[\mathtt{R} \mid \mathbf{t}]\mathbf{X}$$
Image points in normalized coordinates
$$\mathbf{x} = \mathtt{K}[\mathtt{R} \mid \mathbf{t}]\mathbf{X}$$
$$\mathtt{K}^{-1}\mathbf{x} = [\mathtt{R} \mid \mathbf{t}]\mathbf{X}$$
$$\hat{\mathbf{x}} = [\mathtt{R} \mid \mathbf{t}]\mathbf{X} \text{ where } \hat{\mathbf{x}} = \mathtt{K}^{-1}\mathbf{x}$$

Essential Matrix (map X 1st to 2nd camera coord frame)

$$\mathbf{x} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}]\begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}\mathbf{X}$$
$$\mathtt{K}^{-1}\mathbf{x} = [\mathtt{I} \mid \mathbf{0}]\begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}\mathbf{X}$$
$$\hat{\mathbf{x}} = [\mathtt{I} \mid \mathbf{0}]\begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}\mathbf{X} \text{ where } \hat{\mathbf{x}} = \mathtt{K}^{-1}\mathbf{x}$$
$$\hat{\mathbf{x}} = [\mathtt{I} \mid \mathbf{0}]\mathbf{X}_{\text{cam}}$$
$$\hat{\mathbf{x}} = [\mathtt{I} \mid \mathbf{0}]\begin{bmatrix} \tilde{\mathbf{X}}_{\text{cam}} \\ 1 \end{bmatrix}$$
$$\hat{\mathbf{x}} = \tilde{\mathbf{X}}_{\text{cam}} \text{ (up to nonzero scale)}$$

$$\tilde{\mathbf{X}}' = \mathtt{R}\tilde{\mathbf{X}} + \mathbf{t}$$
$$\tilde{\mathbf{X}}' = \mathtt{R}(\lambda\hat{\mathbf{x}}) + \mathbf{t}$$
$$\tilde{\mathbf{X}}' = \lambda\mathtt{R}\hat{\mathbf{x}} + \mathbf{t}$$
$$\lambda\mathtt{R}\hat{\mathbf{x}} + \mathbf{t} = \lambda'\hat{\mathbf{x}}'$$
$$[\mathbf{t}]_\times(\lambda\mathtt{R}\hat{\mathbf{x}} + \mathbf{t}) = [\mathbf{t}]_\times(\lambda'\hat{\mathbf{x}}')$$
$$\lambda[\mathbf{t}]_\times\mathtt{R}\hat{\mathbf{x}} = \lambda'[\mathbf{t}]_\times\hat{\mathbf{x}}'$$
$$\hat{\mathbf{x}}'^\top(\lambda[\mathbf{t}]_\times\mathtt{R}\hat{\mathbf{x}}) = \hat{\mathbf{x}}'^\top(\lambda'[\mathbf{t}]_\times\hat{\mathbf{x}}')$$
$$\lambda\hat{\mathbf{x}}'^\top[\mathbf{t}]_\times\mathtt{R}\hat{\mathbf{x}} = 0$$
$$\hat{\mathbf{x}}'^\top[\mathbf{t}]_\times\mathtt{R}\hat{\mathbf{x}} = 0$$
$$\hat{\mathbf{x}}'^\top\mathtt{E}\hat{\mathbf{x}} = 0 \text{ where } \mathtt{E} = [\mathbf{t}]_\times\mathtt{R}$$

$$\hat{\boldsymbol{\ell}}' = \mathtt{E}\hat{\mathbf{x}} \qquad \hat{\boldsymbol{\ell}}' = \mathtt{E}\hat{\mathbf{x}}$$
$$\hat{\mathbf{x}}'^\top\hat{\boldsymbol{\ell}}' = 0 \quad \hat{\mathbf{x}}'^\top\hat{\boldsymbol{\ell}}' = 0$$
$$\hat{\mathbf{x}}'^\top\mathtt{E}\hat{\mathbf{x}} = 0 \quad \hat{\mathbf{x}}'^\top\mathtt{E}\hat{\mathbf{x}} = 0$$
$$\hat{\mathbf{e}}'^\top\hat{\boldsymbol{\ell}}' = 0 \quad \hat{\mathbf{e}}'^\top\hat{\boldsymbol{\ell}}' = 0$$

Left: Map point from 1[st] image to l' in 2[nd]
Right: Map point from 2[nd] image to l in 1[st]
Epipolar Rectification (don't change camera centers)

$$\tilde{0}_{\text{cam}} = \mathtt{R}\tilde{\mathbf{C}} + \mathbf{t} \qquad \mathbf{x} = \mathtt{K}[\mathtt{R} \mid \mathbf{t}]\mathbf{X} \qquad \tilde{\mathbf{C}} = -\mathtt{R}_{\text{virtual}}^\top\mathbf{t}_{\text{virtual}}$$
$$-\mathtt{R}\tilde{\mathbf{C}} = \mathbf{t} \qquad \mathbf{x} = \mathtt{K}[\mathtt{R} \mid -\mathtt{R}\tilde{\mathbf{C}}]\mathbf{X} \qquad \tilde{\mathbf{C}} = -\mathtt{R}_{\text{real}}^\top\mathbf{t}_{\text{real}}$$
$$\tilde{\mathbf{C}} = -\mathtt{R}^\top\mathbf{t} \qquad \mathbf{x} = \mathtt{K}\mathtt{R}[\mathtt{I} \mid -\tilde{\mathbf{C}}]\mathbf{X} \qquad -\mathtt{R}_{\text{virtual}}^\top\mathbf{t}_{\text{virtual}} = -\mathtt{R}_{\text{real}}^\top\mathbf{t}_{\text{real}}$$
$$\mathbf{t}_{\text{virtual}} = \mathtt{R}_{\text{virtual}}\mathtt{R}_{\text{real}}^\top\mathbf{t}_{\text{real}}$$
$$\mathbf{x}_{\text{real}} = \mathtt{K}_{\text{real}}\mathtt{R}_{\text{real}}[\mathtt{I} \mid -\tilde{\mathbf{C}}]\mathbf{X} \qquad \mathbf{x}_{\text{virtual}} = \mathtt{K}_{\text{virtual}}\mathtt{R}_{\text{virtual}}[\mathtt{I} \mid -\tilde{\mathbf{C}}]\mathbf{X}$$
$$\mathtt{R}_{\text{real}}^\top\mathtt{K}_{\text{real}}^{-1}\mathbf{x}_{\text{real}} = [\mathtt{I} \mid -\tilde{\mathbf{C}}]\mathbf{X} \qquad \mathbf{x}_{\text{virtual}} = \mathtt{K}_{\text{virtual}}\mathtt{R}_{\text{virtual}}\mathtt{R}_{\text{real}}^\top\mathtt{K}_{\text{real}}^{-1}\mathbf{x}_{\text{real}}$$
$$\mathbf{x}_{\text{virtual}} = \mathtt{H}\mathbf{x}_{\text{real}}, \text{ where } \mathtt{H} = \mathtt{K}_{\text{virtual}}\mathtt{R}_{\text{virtual}}\mathtt{R}_{\text{real}}^\top\mathtt{K}_{\text{real}}^{-1}$$

H should minimize image distortion and map epipole e to (1,0,0), point infinity on x-axis.
Forward warping: $x_{virtual}$ = Hx_{real}
Backward warping: x_{real} = $Hx_{virtual}$

Calibrated Stereo and Feature Matching (Part 2)

Two approaches to establishing correspondences
Feature-Based (sparse): For each iamge, process "monocular" image to obtain cures (corners, SIFT, etc.).
Area-Based (dense): Directly compare regions in images.
Match Metric Summary (NCC and NSDD same matches)

Normalized Cross-Correlation (NCC)	$\dfrac{\sum_{u,v}(I_1(u,v) - \bar{I}_1) \cdot (I_2(u+d,v) - \bar{I}_2)}{\sqrt{\sum_{u,v}(I_1(u,v) - \bar{I}_1)^2 \cdot \sum_{u,v}(I_2(u+d,v) - \bar{I}_2)^2}}$
Sum of Squared Differences (SSD)	$\sum_{u,v}(I_1(u,v) - I_2(u+d,v))^2$
Normalized SSD	$\sum_{u,v}\left(\dfrac{(I_1(u,v) - \bar{I}_1)}{\sqrt{\sum_{u,v}(I_1(u,v) - \bar{I}_1)^2}} - \dfrac{(I_2(u+d,v) - \bar{I}_2)}{\sqrt{\sum_{u,v}(I_2(u+d,v) - \bar{I}_2)^2}}\right)^2$
Sum of Absolute Differences (SAD)	$\sum_{u,v}\lvert I_1(u,v) - I_2(u+d,v)\rvert$
Zero Mean SAD	$\sum_{u,v}\lvert(I_1(u,v) - \bar{I}_1) - (I_2(u+d,v) - \bar{I}_2)\rvert$
Rank	$I_k'(u,v) = \sum_{m,n} I_k(m,n) < I_k(u,v)$ $\sum_{u,v}(I_1'(u,v) - I_2'(u+d,v))$
Census	$I_k'(u,v) = BITSTRING_{m,n}(I_k(m,n) < I_k(u,v))$ $\sum_{u,v} HAMMING(I_1'(u,v), I_2'(u+d,v))$

Issues with matching
Epipolar ordering: difficulty of determining the left-to-right or right-to-left correspondence along an epipolar line.
Ambuguity: multiple points in image can match with a point in the other image (should be one-to-one matching).
Window size: size can impact robustness of matching. Small is sensitive to noise. Large may blur details.
Window shape: For example, rectangle suboptimal for matching along curved or non-linear structures. Tilted scenes can project square window into quadrilateral.
Lighting: Does metric handle differences of brightness?
Half occluded regions: regions visible in one camera, but not in the other. They can be a cue for a depth change
Summary of Stereo Constraints

CONSTRAINT	BRIEF DESCRIPTION
1-D Epipolar Search	Arbitrary images of the same scene may be rectified based on epipolar geometry such that stereo matches lie along one-dimensional scanlines. This reduces the computational complexity and also reduces the likelihood of false matches.
Monotonic Ordering	Points along an epipolar scanline appear in the same order in both stereo images, assuming that all objects in the scene are approximately the same distance from the cameras.
Image Brightness Constancy	Assuming Lambertian surfaces, the brightness of corresponding points in stereo images are the same.
Match Uniqueness	For every point in one stereo image, there is at most one corresponding point in the other image.
Disparity Continuity	Disparities vary smoothly (i.e. disparity gradient is small) over most of the image. This assumption is violated at object boundaries.
Disparity Limit	The search space may be reduced significantly by limiting the disparity range, reducing both computational complexity and the likelihood of false matches.
Fronto-Parallel Surfaces	The implicit assumption made by area-based matching is that objects have fronto-parallel surfaces (i.e. depth is constant within the region of local support). This assumption is violated by sloping and creased surfaces.
Feature Similarity	Corresponding features must be similar (e.g. edges must have roughly the same length and orientation).
Structural Grouping	Corresponding feature groupings and their connectivity must be consistent.

Estimate Depth: Given two image measurements x and x', estimate scene point X̂ that minimizes d(x,x̂)[2] + d(x',x̂')[2]
Binocular Stereo: X_L and X_R are known. X and Z aren't. d is baseline while f is focal length. (X_L - X_R) is disparity. X = d*X_L/(X_L - X_R) and Z = d*f/(X_L - X_R).
Challenges and Problems
Photometric issues: specularities, non-Lambertial BRDFs.
Surface structure: lack texture/repeat in horopter bracket.
Geometric ambiguities: hard to get good reconstruction as surfaces turn away. Ex: good match at occluding contour.
Variations on Binocular Stereo
Trinocular Epipolar Constraints: involves three camera views, enforcing geometric relationships among corresponding points in those images. Aids in the accurate estimation and enhances robustness of stereo matching.
Helmholtz Reciprocity: BRDF is the same when swap view and light ρ(θ_{in}, Φ_{in}; θ_{out}, Φ_{out}) = ρ(θ_{out}, Φ_{out}; θ_{in}, Φ_{in})

$$\mathrm{i}_l = \rho(\hat{\mathrm{v}}_r, \hat{\mathrm{v}}_l)\frac{\hat{\mathrm{n}} \cdot \hat{\mathrm{v}}_r}{|o_r - p|^2} \quad = \quad \mathrm{i}_r = \rho(\hat{\mathrm{v}}_l, \hat{\mathrm{v}}_r)\frac{\hat{\mathrm{n}} \cdot \hat{\mathrm{v}}_l}{|o_l - p|^2}$$
$$\left(\mathrm{i}_l\frac{\hat{\mathrm{v}}_l}{|o_l - p|^2} - \mathrm{i}_r\frac{\hat{\mathrm{v}}_r}{|o_r - p|^2}\right) \cdot \hat{\mathrm{n}} = 0 \qquad \begin{pmatrix} i_{l1}\vec{w}_{l1}^\top - i_{r1}\vec{w}_{r1}^\top \\ i_{l2}\vec{w}_{l2}^\top - i_{r2}\vec{w}_{r2}^\top \\ \vdots \end{pmatrix}\hat{\mathbf{n}} = \mathbf{0}$$

Multiple views (3+ pairs) yield matrix constraint equation (must be rank 2). Search for depth where rank constaint is satisfied. Surface normal must lie in null space of matrix.
Reciprocal Images: Typical Dataset
Conventional Stereo: constant brightness & no structure in textureless regions. *Photometric Stereo*: Needs reflectance model. No direct depth estimates. *Helmholtz Stereo*: No assumed reflectance. Gives depth & surface normals.

Uncalibrated Stereo and Feature Extraction

Fundamental Matrix (pixel instead of normalized coords):

$$\hat{\mathbf{x}}'^\top\mathtt{E}\hat{\mathbf{x}} = 0, \text{ where } \hat{\mathbf{x}} = \mathtt{K}^{-1}\mathbf{x} \text{ and } \hat{\mathbf{x}}' = \mathtt{K}'^{-1}\mathbf{x}'$$
$$(\mathtt{K}'^{-1}\mathbf{x}')^\top\mathtt{E}\mathtt{K}^{-1}\mathbf{x} = 0$$
$$\mathbf{x}'^\top\mathtt{K}'^{-\top}\mathtt{E}\mathtt{K}^{-1}\mathbf{x} = 0, \text{ where } \mathtt{K}'^{-\top} = (\mathtt{K}'^{-1})^\top = (\mathtt{K}'^\top)^{-1}$$
$$\mathbf{x}'^\top\mathtt{F}\mathbf{x} = 0, \text{ where } \mathtt{F} = \mathtt{K}'^{-\top}\mathtt{E}\mathtt{K}^{-1}$$

Recap: In calibrated stereo, E is calculated from R and t of 2[nd] camera to 1[st]. In uncalibrated stereo, F is calculated from point correspondences in pixel coordinates.

$$\mathtt{E} = [\mathbf{t}]_\times\mathtt{R} \quad \mathbf{x}'^\top\mathtt{F}\mathbf{x} = 0$$

Left: Map point from 1[st] image to l' in 2[nd] image.
Right: Map point from 2[nd] image to l in 1[st] image.

$$\boldsymbol{\ell}' = \mathtt{F}\mathbf{x} \qquad \boldsymbol{\ell} = \mathtt{F}^\top\mathbf{x}'$$
$$\mathbf{x}'^\top\boldsymbol{\ell}' = 0 \qquad \boldsymbol{\ell}^\top = \mathbf{x}'^\top\mathtt{F}$$
$$\boldsymbol{\ell}^\top\mathbf{x} = 0$$
$$\mathbf{x}'^\top\mathtt{F}\mathbf{x} = 0 \qquad \mathbf{x}'^\top\mathtt{F}\mathbf{x} = 0$$
$$\mathbf{e}'^\top\boldsymbol{\ell}' = 0 \qquad \boldsymbol{\ell}^\top\mathbf{e} = 0$$

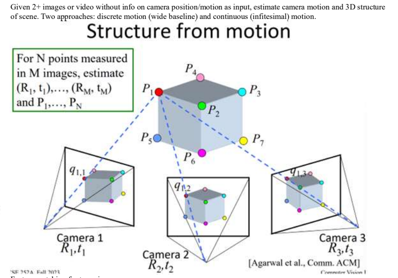
The epipolar constraint must apply to original (x'^{T}Fx = 0) and rectified (x'$^T_{rectified}$$F_{rectified}$$x_{rectified}$ = 0) image pairs. Points in original images map to points in rectified images under the usual rectified transformations.

$$\mathbf{x}_{\text{rectified}} = \mathtt{H}\mathbf{x}$$
$$\mathbf{x}'_{\text{rectified}} = \mathtt{H}'\mathbf{x}'$$
$$\mathbf{x}'^\top_{\text{rectified}}\mathtt{F}_{\text{rectified}}\mathbf{x}_{\text{rectified}} = 0$$
$$(\mathtt{H}'\mathbf{x}')^\top\mathtt{F}_{\text{rectified}}\mathtt{H}\mathbf{x} = 0$$
$$\mathbf{x}'^\top\mathtt{H}'^\top\mathtt{F}_{\text{rectified}}\mathtt{H}\mathbf{x} = 0$$
$$\mathbf{x}'^\top\mathtt{F}\mathbf{x} = 0, \text{ where } \mathtt{F} = \mathtt{H}'^\top\mathtt{F}_{\text{rectified}}\mathtt{H}$$
$$\mathtt{H}'^{-\top}\mathtt{F}\mathtt{H}^{-1} = \mathtt{F}_{\text{rectified}}$$

Fundamental Matrix (more info): 7 DoF. 2 uncalibrated camera projection matrices (up to 3D projective transformation): 24 elements but 22 DoF. 3D projective transformation: 16 elements but 15 DoF. 22-15=7 DoF.
Feature extraction: feature detection and description.
Invariant: value remains unchanged after application of any transformation from the family. Ex: corners are largely invariant to scaling of intensity, except for thresholding.
Covariant: applying any transformation from the set produces the same result in the descriptor. Ex: minor eigenvalues of gradient matrix (corners) are position and orientation covariant but are not scale covariant due to fixed Gaussian filter standard deviation and window size.
Gaussian Pyramid: different levels approximate original image at different scales. Binominal filter kernel approximages kernel of Gaussian with variance = 1 (Pascal's triangle). Colvolve filter each level with kernel, then sample with rate (usually 2) for the next level. Convolution of 2 Gaussians is a Gaussian. Convolution is associative. That is, this whole process is equivalent to convolving two Gaussians with different variances.
Scale Space: formal theory for image structures at different scales. Image represented by 1-parameter family of Gaussian low-pass filtered images. Scale parameter is the variance of the Gaussian filter (use border mirror padding). Details much smaller than the 2X standard deviation removed from the image at that scale parameter.
Gaussian filter meets all scale space axioms: linearity, shift invariance, semi-group structure, rotational symmetry, etc.
Scale Space VS Gaussian Pyramid: pyramid levels (rate=2) correspond to Gaussian kernel with SD of 0, 1, 2, 4, 8… Rate dictates SD of levels. SS doesn't have this imposition; all SDs available. In SS, all images same size, so features located in same coords as OG image. Pyramid must be scaled up to coords of orignal image, imprecise.
SIFT: keypoints (KP) invariant to scale and rotation. Robust to changes in viewpoint, illumination, and noise.
Construct SS → initial KPs → improve location accuracy (interpolate diff values for subpixel coords) → delete unsuitable KPs (low contrast, poorly localized, associated with edges - want corners) → histogram of orientations (36 bins, weighted by magnitude and Gaussian) → descriptors.
Each octave corresponds to 2X SD. First image each new octave is downsampled third image from previous one.
More info about histogram of orientations: create additional keypoint for other bins within 80% size of largest bin. Improve orientation using interpolation – fit parabola to values of largest bin and neighboring bins. Local region about each oriented keypoint is invariant to scale, orientation, illumination, and image viewpoint. Robust to illumination by unitizing and thresholding.

Structure from Motion

Given 2+ images or video without info on camera position/motion as input, estimate camera motion and 3D structure of scene. Two approaches: discrete motion (wide baseline) and continuous (infitesimal) motion.



Feature matching for two views
Greedy algorithm: given feature in one image, find best match in 2[nd] image irrespective of other matches. Suitable for small motions, little rotation, small search window.
Other algorithms: must compare descriptor over rotation. Can't consider all pairings (too many), so do these instead: manual correspondence (photogrammetry), robust outlier rejections (RANSAC), more descriptive features (line segs, SIFT, larger regions, color), use video sequence to track, but perform SFM with 1[st] and last image.
Two view process for calibrated cameras
Input: 2 images (or video frames) → detect feature points → determine correspondences → compute essential matrix → retrieve relative camera rotation and translation (TO SCALE) from E → Optional: perform dense stereo matching using recovered epipolar geometry → reconstruct corresponding 3D scene points (to scale).
Comparing two-view geometry of essential and fundamental matrices
Calibrated and uncalibrated. Normalized and pixel coordinates. Both rank 2 (two nonzero singular values are equal). 5 DoF (rotation and translation, but homogeneous matrix to scale) and 7 DoF (homogeneous matrix to scale and def(F) = 0). Similarity/Euclidian reconstruction and projective reconstruction.
Number of solutions for point correspondences for essential matrix
5 point correspondences → up to 10 (real) solutions
6 point correspondences → 1 solution
7 point correspondences → 1 or 3 real solutions (and 0 or 2 complex ones)
8+ point correspondences → 1 solution
Number of solutions for point correspondences for fundamental matrix
7 point correspondences → 1 or 3 real solutions (and 0 or 2 complex ones)
8+ point correspondences → 1 solution
For CSE 252A, only need to understand that, since E/F have 5/7 DoF, then it can be estimated from 5/7 point correspondences, but there are multiple solutions. Typically, more measurements yield better estimates and less number of solutions.
Essential matrix linear estimation

$$\hat{\mathbf{x}}_i'^\top\mathtt{E}\hat{\mathbf{x}}_i = 0\,\forall i$$
$$\begin{bmatrix}\hat{x}_i' & \hat{y}_i' & \hat{w}_i'\end{bmatrix}\begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix}\begin{bmatrix}\hat{x}_i \\ \hat{y}_i \\ \hat{w}_i\end{bmatrix} = 0$$
$$\hat{x}_i\hat{x}_i'e_{11} + \hat{y}_i\hat{x}_i'e_{12} + \hat{w}_i\hat{x}_i'e_{13} + \hat{x}_i\hat{y}_i'e_{21} + \hat{y}_i\hat{y}_i'e_{22} + \hat{w}_i\hat{y}_i'e_{23} + \hat{x}_i\hat{w}_i'e_{31} + \hat{y}_i\hat{w}_i'e_{32} + \hat{w}_i\hat{w}_i'e_{33} = 0$$
$$\begin{bmatrix}\hat{x}_i\hat{x}_i' & \hat{y}_i\hat{x}_i' & \hat{w}_i\hat{x}_i' & \hat{x}_i\hat{y}_i' & \hat{y}_i\hat{y}_i' & \hat{w}_i\hat{y}_i' & \hat{x}_i\hat{w}_i' & \hat{y}_i\hat{w}_i' & \hat{w}_i\hat{w}_i'\end{bmatrix}\begin{bmatrix} e_{11} \\ e_{12} \\ e_{13} \\ e_{21} \\ e_{22} \\ e_{23} \\ e_{31} \\ e_{32} \\ e_{33} \end{bmatrix} = 0$$
$$\mathbf{a}_i = (\hat{x}_i\hat{x}_i', \hat{y}_i\hat{x}_i', \hat{w}_i\hat{x}_i', \hat{x}_i\hat{y}_i', \hat{y}_i\hat{y}_i', \hat{w}_i\hat{y}_i', \hat{x}_i\hat{w}_i', \hat{y}_i\hat{w}_i', \hat{w}_i\hat{w}_i')^\top$$
$$\mathbf{e} = (e_{11}, e_{12}, e_{13}, e_{21}, e_{22}, e_{23}, e_{31}, e_{32}, e_{33})^\top \qquad \mathbf{a}_i^\top\mathbf{e} = 0$$

Given $n \geq 8$ point correspondences

$$\mathbf{e} = (e_{11}, e_{12}, e_{13}, e_{21}, e_{22}, e_{23}, e_{31}, e_{32}, e_{33})^\top$$
$$\mathtt{A} = \begin{bmatrix}\mathbf{a}_1^\top \\ \mathbf{a}_2^\top \\ \vdots \\ \mathbf{a}_n^\top\end{bmatrix} \qquad \begin{bmatrix}\mathbf{a}_1^\top \\ \mathbf{a}_2^\top \\ \vdots \\ \mathbf{a}_n^\top\end{bmatrix}\mathbf{e} = \mathbf{0}$$
$$\mathtt{A}\mathbf{e} = \mathbf{0}, \text{ solve for } \mathbf{e}$$
$$\mathbf{a}_i = (\hat{x}_i\hat{x}_i', \hat{y}_i\hat{x}_i', \hat{w}_i\hat{x}_i', \hat{x}_i\hat{y}_i', \hat{y}_i\hat{y}_i', \hat{w}_i\hat{y}_i', \hat{x}_i\hat{w}_i', \hat{y}_i\hat{w}_i', \hat{w}_i\hat{w}_i')^\top$$

$\mathtt{A}\mathbf{e} = \mathbf{0}$, solve for $\mathbf{e}$ using singular value decomposition (SVD)
$$\mathtt{A} = \mathtt{U}\Sigma\mathtt{V}^\top$$
where
$\mathtt{U}$ and $\mathtt{V}$ are orthogonal matrices
$\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_9)$, where $\sigma_i \geq \sigma_{i+1} \geq 0$
Columns of $\mathtt{U}$ are left singular vectors corresponding to singular values $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_9)^\top$ and columns of $\mathtt{V}$ are right singular vectors corresponding to singular values.
$\mathbf{e} = (e_{11}, e_{12}, e_{13}, e_{21}, e_{22}, e_{23}, e_{31}, e_{32}, e_{33})^\top$ is right singular vector corresponding to smallest singular value (i.e., $\mathbf{e}$ is the last column of $\mathtt{V}$)

$$\mathtt{E} = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix}$$

then enforce constraints (rank 2 and other two singular values are equal)
$$\mathtt{E} = \mathtt{U}\Sigma\mathtt{V}^\top$$
$$\mathtt{E} = \mathtt{U}\Sigma'\mathtt{V}^\top, \text{ where } \Sigma' = \operatorname{diag}(1, 1, 0)$$

Retrieve relative camera rotaiton and direction of translation from essential matrix E = [t]xR where P̂ = [I | 0] and P̂' = [R | t]. There are four solutions for R and t, but only one reconstructs the point in front of both cameras.
Fundamental matrix linear estimation

$$\mathbf{x}_i'^\top\mathtt{F}\mathbf{x}_i = 0\,\forall i$$
$$\begin{bmatrix}x_i' & y_i' & w_i'\end{bmatrix}\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}\begin{bmatrix}x_i \\ y_i \\ w_i\end{bmatrix} = 0$$
$$x_ix_i'f_{11} + y_ix_i'f_{12} + w_ix_i'f_{13} + x_iy_i'f_{21} + y_iy_i'f_{22} + w_iy_i'f_{23} + x_iw_i'f_{31} + y_iw_i'f_{32} + w_iw_i'f_{33} = 0$$

Warning: data normalization must be used!

$$\begin{bmatrix}x_ix_i' & y_ix_i' & w_ix_i' & x_iy_i' & y_iy_i' & w_iy_i' & x_iw_i' & y_iw_i' & w_iw_i'\end{bmatrix}\begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$
$$\mathbf{a}_i = (x_ix_i', y_ix_i', w_ix_i', x_iy_i', y_iy_i', w_iy_i', x_iw_i', y_iw_i', w_iw_i')^\top$$
$$\mathbf{f} = (f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33})^\top \qquad \mathbf{a}_i^\top\mathbf{f} = 0$$

Given $n \geq 8$ point correspondences

$$\mathbf{f} = (f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33})^\top$$
$$\mathtt{A} = \begin{bmatrix}\mathbf{a}_1^\top \\ \mathbf{a}_2^\top \\ \vdots \\ \mathbf{a}_n^\top\end{bmatrix} \qquad \begin{bmatrix}\mathbf{a}_1^\top \\ \mathbf{a}_2^\top \\ \vdots \\ \mathbf{a}_n^\top\end{bmatrix}\mathbf{f} = \mathbf{0}$$
$$\mathtt{A}\mathbf{f} = \mathbf{0}, \text{ solve for } \mathbf{f}$$
$$\mathbf{a}_i = (x_ix_i', y_ix_i', w_ix_i', x_iy_i', y_iy_i', w_iy_i', x_iw_i', y_iw_i', w_iw_i')^\top$$

$Af = 0$, solve for f using singular value decomposition (SVD)

$$A = U\Sigma V^\top$$

where

U and V are orthogonal matrices
$\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_9)$, where $\sigma_i \geq \sigma_{i+1} \geq 0$

Columns of U are left singular vectors corresponding to singular values $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_9)^\top$ and columns of V are right singular vectors corresponding to singular values.

$f = (f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33})^\top$ is right singular vector corresponding to smallest singular value (i.e., f is the last column of V)

$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

Warning: data normalization must be used!

then enforce constraint (rank 2)

$F = U\Sigma V^\top$, where $\Sigma = \text{diag}(\sigma_1, \sigma_2, \sigma_3)$

$F' = U\Sigma' V^\top$, where $\Sigma' = \text{diag}(\sigma_1, \sigma_2, 0)$

Retrieve uncalibrated camera projection matrices from fundamental matrix.

$$P = [I \mid 0] \text{ and } P' = [[e']_\times F + e' v^\top \mid \lambda e'],$$

where v is any 3-vector and λ is a non-zero scalar.

$$\text{If } v = 0 \text{ and } \lambda = 1, \text{ then } P' = [[e']_\times F \mid e'].$$

Retrieve epipoles from fundamental matrix.

$Fe = 0$ $\quad e$ is (right) null space of F

$e'^\top F = 0^\top$ $\quad e'$ is left null space of F

$(F^\top e' = 0$ $\quad e'$ is (right) null space of $F^\top)$

Three Views

Trifocal Tensor: 3x3x3 tensor. 27 elements, 18 DoF. 33 DoF (3 camera projection matrices) minus 15 DoF (3D projective transformation). Uses tensor notation. Retrieves camera projection matrices. Can do point-point-point, point-line-point (image line must pass through corres. image point), point-line-line (image lines don't need to correspond but must pass through corres. image points), and line-line-line. In N-view structure from motion, minimize g(P,R,T) with respect to P,R,T non-linear least squares.

Calibrated structure from motion

Consider m images of n points, how many unknowns? 3D structure: 3n. Cameras: 6m-7. Total: 3n+6m-7. Measurements: 2mn. Solution when 3n+6m-7<=2mn.

Uncalibrated structure from motion

Consider m images of n points, how many unknowns? 3D structure: 3n. Cameras: 11m-15. Total: 3n+11m-15. Measurements: 2mn. Solution when 3n+11m-15<=2mn.

N-View Geometry Reconstruction

Bundle adjustment: simultaneous adjustment of parameters for all cameras and all 3D scene points. Minimize reprojection error in all images. Reconstruction of cameras and 3D scene points to similarity (calibrated) or projective (uncalibrated) ambiguity.

$$\min_{\hat{P}^i, \hat{X}_j} \sum_{ij} d(\hat{P}^i \hat{X}_j, x_j^i)^2$$

Minimize sum of squared reprojection errors

$$g(\mathbf{P},\mathbf{R},\mathbf{T}) = \sum_{i=1}^{M}\sum_{j=1}^{N} w_{ij} \left\| P(P_i, \mathbf{R}_j, \mathbf{t}_j) - \begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix} \right\|^2$$

indicator variable: 1: If point i is visible in image j; 0: Otherwise
predicted image location
observed image location

g(P,R,T) is optimized with non-linear least squares. Levenberg-Marquardt is a popular choice: It combines aspects of both gradient descent and Gauss-Newton methods to minimize objective function. Practical challenges include initialization (Bootstrap from 2 View Solution between pairs and outliers).

Initial conditions for optimization: perform 2 View SFM for image pairs that have 8+ matching pairs of features. This forms a graph. Can propagate translatin and orientation across a spanning tree of a graph. Can be more accurate if you use full graph.

Model Fitting

Examples: segment linked edge chains into curve features (i.e., line segments), group unlinked or unrelated edges into lines (or curves), accurately fitting parametric curves (e.g., lines) to grouped edge points.

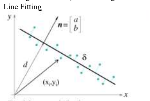
Hough Transform (HT): Typically uses a d = x*cos(θ) + y*sin(θ) as the parameterization. D is perpendicular distance from the line to origin. θ is the angle this perpendicular makes with the x-axis. In each point in parametric form $y=f(x,a_1,a_2,\ldots,a_p)$ or $g(x,a_1,a_2,\ldots,a_p)=0$. $a_1,a_2,\ldots,a_p$ are the parameters. The parameter space if p-dimensional. The accumulating array is large. Example: Finding circles using $(x-x_c)^2+(y-y_c)^2=r^2$ where the parameters are the circle's center (x_c,y_c) and radius r. Three dimensional generalized Hough space. Given an edge point (x_i,y_i), compute r, and increment $H(x_c,y_c,r)$.

Basic Hough Transform Algorithm: Initialize H[d,θ]=0 (accumulator array). For each edge point I[x,y] in the image, compute d = x*cos(θ) + y*sin(θ) for θ = 0 to 180 and increment H[d,θ]. Find values of (d,θ) where H[d,θ] is global maximum. The detected line in the image is given by d = x*cos(θ) + y*sin(θ).

Transmission Electron Microscopy (TEM) Image of Keyhole Limpet Hemocyanin (KLH) with detected particles (recall the cylinders that are circles on one side and rectangles on the other): Use Canny edge detection – a sequence of ordered HTs is applied in order from computationally simplest one to the most complex one. Edges covered by detected shapes removed immediately from images following last HT.

Image → Edge detection → Circle detection (move circle coords to candidate particles) → Remove circles → Rectangle detection (move rectangle coords to candidate particles) → Candidate particles.

Line Fitting



Given n points (x_i, y_i), estimate parameters of line $ax_i + by_i - d = 0$ subject to the constraint that $a^2 + b^2 = 1$.

Note: $ax_i + by_i - d$ is distance from (x_i, y_i) to line.

Problem: minimize

$$E(a,b,d) = \sum_{i=1}^{n}(ax_i + by_i - d)^2$$

Cost Function: Sum of squared distances between each point and the line

with respect to (a,b,d).

1. Minimize E with respect to d:

$$\frac{\partial E}{\partial d} = 0 \Rightarrow d = \frac{1}{n}\sum_{i=1}^{n} ax_i + by_i = a\bar{x} + b\bar{y}$$

Where $(\bar{x}, \bar{y})$ is the mean of the data points

2. Substitute d back into E

$$E = \sum_{i=1}^{n}[a(x_i - \bar{x}) + b(y_i - \bar{y})]^2 = \|U\mathbf{n}\|^2 \quad \text{where} \quad U = \begin{pmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \ldots & \ldots \\ x_n - \bar{x} & y_n - \bar{y} \end{pmatrix}$$

where $\mathbf{n}=(a, b)^T$.

3. Minimize $E=|U\mathbf{n}|^2=\mathbf{n}^T U^T U \mathbf{n}=\mathbf{n}^T S \mathbf{n}$ with respect to a, b subject to the constraint $\mathbf{n}^T\mathbf{n} = 1$. Note that S is given by

$$S = U^T U = \begin{pmatrix} \sum_{i=1}^{n} x_i^2 - n\bar{x}^2 & \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y} \\ \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y} & \sum_{i=1}^{n} y_i^2 - n\bar{y}^2 \end{pmatrix}$$

which is real, symmetric, and positive definite

4. This is a constrained optimization problem in $\mathbf{n}$. Solve with Lagrange multiplier

$$L(\mathbf{n}) = \mathbf{n}^T S \mathbf{n} - \lambda(\mathbf{n}^T\mathbf{n} - 1)$$

Take partial derivative (gradient) w.r.t. $\mathbf{n}$ and set to 0.

$$\nabla L = 2S\mathbf{n} - 2\lambda\mathbf{n} = 0$$

or

$$S\mathbf{n} = \lambda\mathbf{n}$$

$\mathbf{n}=(a,b)$ is an Eigenvector of the symmetric matrix S (the one corresponding to the smallest Eigenvalue).

5. d is computed from Step 1.

RANSAC for line fitting: Given S points and threshold τ, determine best fit line in presence of outliers. Repeat this N times: Select 2 points randomly, determine line equation from 2 points, compute "support" – number of points within distance τ from line (inliers). Line with largest "support" wins.

RANSAC more generally (what do we need to apply it?): A parameterized model, a way to estimate the model parametesr from s data points $\{x_1,\ldots,x_s\}$, given parameters of model, a way to estimate distance from data point x_i to model.

Algorithm

REPEAT

(i) Randomly select a sample of s data points from S

(ii) Instantiate the model from this sample.

(iii) Determine the set of data points S_i which are within a distance threshold t of the model. The set S_i is the **consensus set** of samples and defines the inliers of S.

(iv) $S_{largest} = S_i$ if S_i is larger than $S_{largest}$

UNTIL (The size of S_i is greater than some threshold T) OR (There have been N samples)

The model is re-estimated using all the points in $S_{largest}$

RANSAC for fundamental matrix: 8 samples/matched pairs (SIFT) for each image. Compute distance using L^2 distance of point to epipolar line, epipolar constraint, reprojection error or its Sampson error approximation.

Distance equations: Dist1(q,q') = dist(q, L(q')) + dist(q', L'(q)) or Dist2(q,q') = $q'^T F q$. Note: F must be normalized, perhaps by dividing by ||F||.

How many samples needed to be confident that you found only inliers? Choose N (number of samples) so that, with probability p (usually p=0.99), at least one of N random samples is free from outliers. s = sample size (i.e., number of points needed for the model). e = proportion of outliers in the data.

RANSAC for estimating F: 8 point algorithm requires 8 correctly matched pairs. Hypothetically, if matching method using SIFT leads to 40% incorrect pairs, then chance of selecting 8 correct pairs is $(1-0.4)^8$=0.016. To be 99% sure at least one random group of 8 pairs has only correct matches, might need to randomly draw 272 pairs.

Formula: $(1-(1-e)^s)^N = 1-p$. Solve for $N = \log(1-p)/\log(1-(1-e)^s)$.

p: probability that at least one of N samples contains no outliers.

1-p: probability that all N samples contain outliers.

s: sample size (i.e., number of points needed for model).

e: proportion of outliers in the data.

1-e: probability of a data sample being an inlier.

$(1-e)^s$: probability that s samples are all inliers.

$1-(1-e)^s$: probability that s samples have at least one outlier.

$(1-(1-e)^s)^N$: probability that N samples all have outliers.

Distance threshold: choose threshold τ so probability for inlier is α (usually 0.95 and empirically). Zero-mean Gaussian noise σ then $d_\perp^2$ follows X_m^2 distribution with m = codimension of model (codimension of subspace = dim of space – dim of subspace).

Number of inliers threshold: typically, terminate when inlier ratio reaches the expected ratio of inliers. T = (1-e)N

Optical Flow and Motion

Motion Field: The motion field is the projection of the 3D scene motion into the image. Causes: camera moves (rotates or translates), objects in scene move, objects articulate (pliers, humans, animals), objects bend and deform (fish), blowing smoke/clouds, etc. Example motion field: Camera moving straight along optical axis. The "instantaneous" velocity of all points in an image LOOMING. Green arrow points to the Focus of Expansion (FOE): Intersection of velocity vector with the image plane.

With just this information, we can calculate 1. Direction of motion. 2. Time to collision.

Rigid motion: General Case: $\dot{X} = -(\omega \times X + t)$ where t is velocity vector and ω is angular velocity vector. NOTE: t IS THE VELOCITY VECTOR, NOT v LIKE USUAL.

General Motion:

$$\text{Pixel coordinates} \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z}\begin{bmatrix} X \\ Y \end{bmatrix}$$

Taking the time derivative of both sides yields

$$\text{Motion field pixel coordinates} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{f}{Z}\begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} - \frac{f\dot{Z}}{Z^2}\begin{bmatrix} X \\ Y \end{bmatrix}$$

$$\text{Substitute } \dot{\mathbf{X}} = -(\boldsymbol{\omega} \times \mathbf{X} + \mathbf{t})$$

$$\text{Then, substitute } \quad x = \frac{f}{Z}X \quad y = \frac{f}{Z}Y$$

Motion Field Equation

$$\dot{x} = \frac{t_3 x - t_1 f}{Z} - \omega_2 f + \omega_3 y + \frac{\omega_1 xy - \omega_2 x^2}{f}$$

$$\dot{y} = \frac{t_3 y - t_2 f}{Z} + \omega_1 f - \omega_3 x + \frac{\omega_1 y^2 - \omega_2 xy}{f}$$

translation only rotation only rotation only rotation only rotation only

Pure Translation: camera is just translating with no rotation (ω=0). ẋ, ẏ is inversely proportional to Z (remember Euclid). FOE is located at (x,y) where ẋ=0, ẏ=0. Substitute and solve for (x,y). Insight: The FOE is the perspective projection of the linear velocity.

If radial about FOE (t_3 is not 0), then motion arrows point toward or away from a point. If t_3=0, then motion is parallel to image plane (FOE point at infinity).

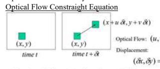
Pure rotation: camera is only rotating (t=0). Motion is independent of t and Z. Only a funciton of x, y, f, and ω. Ex: if ω = $(0,0,1)^T$, then arrows rotate counter-clockwise.

Estimate Depth: If t, ω, and f are known, then for each image point (x,y), solve for depth Z given measured motion u, v at (x,y). Depth inversely proportional to image velocity.

Estimating motion field from images: 1. Feature-based – detect (corner-like) features in image and search for same features nearby. 2. Differential techniques.

Optical Flow (OF): pattern of apparent motion of objects, surfaces, and edges in visual scene caused by relative motion between observer and a scene. Ideally, OF is projection of 3D velocity vectors on the image (i.e., the motion field), but it is not. Estimate pixel motion by finding pixel correspondences (same color) in both images. Assumes 1. Color constancy – point in H(x,y) looks "same" in I(x,y). For grayscale, this is brightness constancy 2. Small motion – points don't move very far.

Optical Flow Constraint Equation



1. Assume brightness of patch remains same in both images:

$$I(x + u\,\delta t, y + v\,\delta t, t + \delta t) = I(x, y, t)$$

2. Assume small motion: (Taylor expansion of LHS up to first order)

$$I(x,y,t) + \delta x\frac{\partial I}{\partial x} + \delta y\frac{\partial I}{\partial y} + \delta t\frac{\partial I}{\partial t} = I(x,y,t)$$

3. Subtracting $I(x,y,t)$ from both sides and dividing by δt

$$\frac{\delta x}{\delta t}\frac{\partial I}{\partial x} + \frac{\delta y}{\delta t}\frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} = 0$$

4. Assume small interval, this becomes:

$$\frac{dx}{dt}\frac{\partial I}{\partial x} + \frac{dy}{dt}\frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} = 0$$

$$\frac{dx}{dt}\frac{\partial I}{\partial x} + \frac{dy}{dt}\frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} = 0$$

We can measure $\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}, \frac{\partial I}{\partial t}$

- $\frac{\partial I}{\partial x}$ Filter image with kernel [-1, 0, 1]
- $\frac{\partial I}{\partial y}$ Filter image with $[-1, 0, 1]^T$
- $\frac{\partial I}{\partial t}$ Consider stacking 3 images at (t-1, t, t+1), then filter over time with kernel [-1,0,1]; or backward difference

We want to solve for $\frac{dx}{dt}, \frac{dy}{dt}$

One equation, two unknowns → Cannot solve it

Apeture Problem & Normal Flow: When window size too small, motion estimation becomes increasingly ambiguous.

We measure:

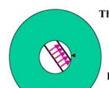
$$I_x = \frac{\partial I}{\partial x}, \quad I_y = \frac{\partial I}{\partial y}, \quad I_t = \frac{\partial I}{\partial t}$$

The gradient constraint:

$$I_x u + I_y v + I_t = 0$$

$$\nabla I \cdot \vec{U} = 0$$

Defines a line in the (u,v) space

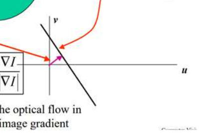


We want to estimate Flow vector

$$u = \frac{dx}{dt}, \quad v = \frac{dy}{dt}$$

Normal Flow:

$$u_\perp = -\frac{I_t}{|\nabla I|}\frac{\nabla I}{|\nabla I|}$$



The component of the optical flow in the direction of the image gradient

Two ways to get flow

1. Think globally, regularize over image. 2. Look over window, assume constant motion.

Horn and Schunk Algorithm (minimize $e_s+\lambda e_c$)

Additional smoothness constraint :

$$e_s = \iint ((u_x^2 + u_y^2) + (v_x^2 + v_y^2))dxdy,$$

besides OF constraint equation term

$$e_c = \iint (I_x u + I_y v + I_t)^2 dxdy,$$

Lucas-Kanade: Integrate over a patch

Assume a single velocity (u,v) for pixels within an image patch Ω.

$$E(u,v) = \sum_{x,y \in \Omega} \left(I_x(x,y)u + I_y(x,y)v + I_t\right)^2$$

$$\frac{dE(u,v)}{du} = \sum 2I_x\left(I_x u + I_y v + I_t\right) = 0$$

$$\frac{dE(u,v)}{dv} = \sum 2I_y\left(I_x u + I_y v + I_t\right) = 0$$

In matrix form:

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}\begin{pmatrix} u \\ v \end{pmatrix} = -\begin{pmatrix} \sum I_x I_t \\ \sum I_y I_t \end{pmatrix}$$

On the LHS: sum of the 2x2 outer product tensor of the gradient vector

$$\left(\sum \nabla I \nabla I^T\right)\vec{U} = -\sum \nabla I I_t$$

E(u,v) is minimized when partial derivatives equal zero. So, the optical flow U=(u,v) can be written as MU=b and optical flow is just $U=M^{-1}b$.

Singularities & Apeture Problem: M is singular if constant brightness in image ∇I=0, window is one pixel, along an edge (where the direction of ∇I is the same [or zero] in the window), aperture problem still exists. M is full rank for regions of bidirectional texturedness (e.g., corners).

Some variants

Iterative refinement: Estimate velocity at each pixel using one iteration of L&K. Warp one image toward other using estimated flow field. Refine by repeating.

Coarse to fine (image pyramids): multi-resolution approach involving Gaussian pyramids of input images. OF initially estimated at lower resolutions. Solution is refined at higher resolutions (warp & upsample using flow from previous level).

Local/global motion models (Parametric Motion Models): 2D models (translation) – affine, quadratic, and planar projective transform (Homography). 3D models: instantaneous camera motion models, homography+epipole, plane+parallax.

Robust estimation: No information on this.

Visual Tracking: OF is pixel-level tracking, but we now consider tracking objects. Main challenges: 3D pose variation, target occlusion, illumination variation, camera jitter, expression variation. Main tracking notions below:

State $\boldsymbol{\phi}$(t): finite number of parameters (vector) characterizing the "state" (location, size, pose, etc. of thing being tracked).

Dynamics: How does the state change over time? How is that change constrained?

Prediction: Given state at time t, what is estimate at t+1? Use state and dynamics.

Representation: How do you represent the thing being tracked?

Data Association: What measurements correspond to which object?

Correction: Given the predicted state and measurement at time t+1, updated state $\boldsymbol{\phi}_c(t+1) = f(\boldsymbol{\phi}_P(t+1), M(t+1))$.

Recognition, Detection, and Classification (Part 1)

Given database of objects and an image, determine if the objects are present in image, where they are, how many there are, and what their 3D pose is.

Recognition Challenges

Within-class variability: different objects within the class have different shapes/material characteristics, deformable/articulated/compositional.

Pose variability: 2D image transformation (translation, rotation, scale), 3D pose variability and perspective projection.

Lighting: direction (multiple sources and type), color, and shadows.

Occlusion: partial or self occlusion.

Clutter in background: false positives.

Binary Classifier

		Classifier Output	
		Same	Different
Truth	Same	True Positive Rate	False Negative Rate
	Different	False Positive Rate	True Negative Rate

True Positive Rate (TPR) + False Negative Rate (FNR) = 1

False Positive Rate (FPR) + True Negative Rate (TNR) = 1

ROC Curve: Evaluating a binary classifier

Most classifiers have threshold that can vary - leads to ROC Curve (Receiver Operator Characteristic) – Plot tradeoff between FPR and TPR as a result of varying threshold.

Bayesian Classification: ω is random variable. P(ω) is prior probability. $P(\omega_1)+P(\omega_2)=1$ is exclusivity & exhaustivity. Decide ω_1 if $P(\omega_1) > P(\omega_2)$. x is vector of features. Likelihood $P(x|\omega_1)$ describes feature distribution of ω_1 population. $P(\omega_1|x)$ is posterior. Bayes rule is $P(\omega_1|x)=P(x|\omega_1)*P(\omega_1)/P(x)$. Classify by choosing largest posterior. The challenge is estimating $P(x|\omega_1)$ because need many samples for higher dim x. Instead, can use parametric form for $P(x|\omega_1)$, KNN, learn decision ruls directly, reduce dim of x.

Decision region and decision boundary:

$$R_i = \{\mathbf{x} : g_i(\mathbf{x}) > g_j(\mathbf{x}), \forall j, i \neq j\} \qquad H_{ij} = \{\mathbf{x} : g_i(\mathbf{x}) = g_j(\mathbf{x}), g_i(\mathbf{x}) > g_k(\mathbf{x}), \forall k, i \neq k\}$$

Confusion matrix: TP diagonal, FN along row (excluding diagonal), FP along column (excluding diagonal), recall=TP/(TP+FN), precision=TP/(TP+FP).

Plug-in Classifier: Think of "plugging in" pre-defined algorithm to predict on new data. Once trained, it can be applied directly to classify new instances without further adjustments to the model parameters. Assume class conditional distributions $P(\omega_1|x)$ have parametric form with parameters W. Given training samples x for class ω_1, estimate parameters W. Common parametric form: uniform over region, normal distribution with shared covariance matrix (different means), normal distribution but with different covariance matrices.

Support Vector Machines: Bayes/generative approaches try to model posterior. SVMs try to obtain decision boundary directly. Potentially easier because only need to encode geometry of boundary, not any irrelevant wiggles in posterior. Goal: find hyperplane that divides S into 2 classes. d_i is distance of point to SVM. 4^{th} equation is closest point.

S is separable if $\exists w \in R^n, b \in R$ $\quad y_i(w^T x_i + b) \geq 1$

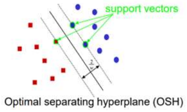
$$d_i = \frac{w^T x_i + b}{w} \qquad w = \|w\| \qquad y_i d_i = \frac{1}{w}$$

Separating hyperplanes $w^T x + b = 0$

Optimal separating hyperplane maximizes 1/w.

$$\text{Minimize} \quad \tfrac{1}{2} w^T w$$

$$\text{Subject to} \quad y_i(w^T x_i + b) \geq 1, \; i = 1,2,\ldots,N$$



Optimal separating hyperplane (OSH)

Solve using Lagrange multipliers on left.

Lagrangian

$$L(w,b,\alpha) = \tfrac{1}{2}w^T w - \sum_{i=1}^{N}\alpha_i\{y_i(w^T x_i + b) - 1\}$$

– at solution

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{N} y_i \alpha_i = 0$$

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N}\alpha_i y_i x_i = 0$$

$$L(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j x_i . x_j$$

– therefore

$$\alpha \geq 0$$

Decision function is $f(x) = \text{sign}(w^T x+b)$. Note: globally optimal solution can always be obtained (convex problem).

Bag-of-features steps: Extract features, learn "visual vocabulary," quantize features using vocabulary, represent images by frequencies (histogram) of "visual words," recognition using histograms as input to classifier.

Pattern classification summary:

Supervised: KNN, Bayesian, Neural Network, SVMs, Kernel methods.

Unsupervised: Clustering, Reinforcement learning.

Recognition, Detection, and Classification (Part 2)

Curse of Dimensionality: What is a linear subspace?

Let V be vector space and W be subset of V. W is subspace iff: Null vector 0 in W. Linear combination of u,v∈W is ∈W too. c*u in W if c is scalar. A k-dim subspace spanned by k lin. ind. vectors/spanned by k-dim orthogonal basis. A d-pixel image $x \in R^d$ can be projected to low-dim feature space $y \in R^k$ by y=Wx where W is KxD matrix. Each training/test image projected to subspace. Recognition performed on R^k too.

How to choose a good W? Drop dims (feature selection), random projections, Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Independent Component Analysis, non-linear dimensionality reduction.

PCA: Assume we have set of n feature vectors x_i (i = 1,…,n) in R^d. Write $u_x = 1/n*\Sigma x_i$ (mean) and $\Sigma_x = 1/(n-1)*\Sigma(x_i-u)*(x_i-u)^T$. Eigen decomposition of covariance matrix $\Sigma_x=V\Lambda V^T$ where Σ_x is positive semidefinite nxn matrix, V is an nxn orthogonal matrix, Λ is diag($\lambda_1,\ldots,\lambda_n$), where $\lambda_i >= \lambda_{i+1} >= 0$. The columns of V are eigenvectors (AKA principal component coefficients) corresponding to eigenvalues $\lambda=(\lambda_1,\ldots,\lambda_n)^T$ (AKA principal component variances). PCA identifies a collection of linear features that are independent, and capture as much variance as possible from a dataset.

Relationship between SVD and eigen decomposition

$$A = U\Sigma V^\top \qquad\qquad A = U\Sigma V^\top$$

$$A^\top A = (U\Sigma V^\top)^\top U\Sigma V^\top \qquad AA^\top = U\Sigma V^\top(U\Sigma V^\top)^\top$$

$$A^\top A = V\Sigma^\top U^\top U\Sigma V^\top \qquad AA^\top = U\Sigma V^\top V\Sigma^\top U^\top$$

$$A^\top A = V\Sigma^\top \Sigma V^\top \qquad AA^\top = U\Sigma\Sigma^\top U^\top$$

$$A^\top A = V\Lambda V^\top, \text{ where } \Lambda = \Sigma^\top\Sigma \qquad AA^\top = U\Lambda U^\top, \text{ where } \Lambda = \Sigma\Sigma^\top$$

where

U and V are orthogonal matrices

$\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$, where $\sigma_i \geq \sigma_{i+1} \geq 0$

$\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$, where $\lambda_i \geq \lambda_{i+1} \geq 0$

variances $\lambda_i = \sigma_i^2 \, \forall i$

$X \in \mathbb{R}^{m \times n}$ $\quad m$ observations, n variables

$$X = \begin{bmatrix} \mathbf{x}^{1\top} \\ \mathbf{x}^{2\top} \\ \vdots \\ \mathbf{x}^{m\top} \end{bmatrix} \quad \hat{X} = \begin{bmatrix} \mathbf{x}^{1\top} - \boldsymbol{\mu}_x^\top \\ \mathbf{x}^{2\top} - \boldsymbol{\mu}_x^\top \\ \vdots \\ \mathbf{x}^{m\top} - \boldsymbol{\mu}_x^\top \end{bmatrix} \quad \boxed{\Sigma_{\mathbf{x}} = \frac{1}{m-1}\hat{X}^\top\hat{X}}$$

SVD of (mean-derivation form of) data matrix

$$\hat{X} = UDV^\top$$

Note: economy SVD can be used

where

$D = \text{diag}(\sigma_1, \ldots, \sigma_{\min(m,n)})$, where $\sigma_i \geq \sigma_{i+1} \geq 0$

columns of V (rows of $V^\top$) are principal component coefficients of $\hat{X}$

Projection of $\hat{X}$ to principal component axes

$\hat{A} = \hat{X}V$ $\quad$ (forward) projection to principal component scores

$\hat{A}V^\top = \hat{X}$ $\quad$ back projection

Dimensionality reduction

Columns of V corresponding to smallest singular values can be removed

$\hat{A}' = \hat{X}V'$ $\quad$ (forward) projection to principal component scores

$\hat{A}'V'^\top = \hat{X}'$ $\quad$ back projection (with loss of data)

SVD Properties

r = rank(A) = number of non-zero singular values. U,V give the orthonormal bases for subspaces of A.

1^{st} r columns of U: Column space of A. Last m-r columns of U: Left nullspace of A.

1^{st} r columns of V: Row space of A. 1^{st} n-r columns of V: (Right) nullspace of A.

For some d<=r, first d cols of U provide best d-dim basis for cols of A in least squares.

PCA for Recognition

Modeling: Given collection of n training images, represent each one as d-dim column vector. Compute mean image and covariance matrix. Compute k eigenvectors of covariance matrix corresponding to the k largest eigenvalues and form matrix $W^T=[u_1,\ldots,u_k]$ (or use SVD). Note that eigenvectors are images. Project training images to k-dim eigenspace. $y_i=Wx_i$.

Recognition: Given test image x, project vectorized image to eigenspace y=Wx. Perform classification of y to the projected training images.

Issues with PCA

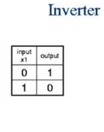
Projection may suppress important detail (smallest variance directions may still be important). Method doesn't take discriminative task into account: typically want to compute features that allow good discrimination, which is not same as largest variance or minimizing reconstruction error.

Fisher's Linear Discriminant

Between-class scatter $\quad$ Within-class scatter $\quad$ Total scatter

$$S_B = \sum_{i=1}^{c}|\chi_i|(\mu_i - \mu)(\mu_i - \mu)^T \quad S_W = \sum_{i=1}^{c}\sum_{x_k \in \chi_i}(x_k - \mu_i)(x_k - \mu_i)^T \quad S_T = \sum_{i=1}^{c}\sum_{x_k \in \chi_i}(x_k - \mu)(x_k - \mu)^T = S_B + S_W$$

Num classes = c, u_i is mean of class X_i, $|X_i|$ is number of samples X_i. If the data points x_i are projected by $y_i=Wx_i$ and the scatter of x_i is S, then the scatter of the projected points y_i is $W^T SW$.



- PCA (Eigenfaces)

$$W_{PCA} = \arg\max_W |W^T S_T W|$$

Maximizes projected total scatter

- Fisher's Linear Discriminant

$$W_{fld} = \arg\max_W \frac{|W^T S_B W|}{|W^T S_W W|}$$

Maximizes ratio of projected between-class to projected within-class scatter

$$W_{opt} = \arg\max_W \frac{|W^T S_B W|}{|W^T S_W W|} = [\mathbf{w}_1 \; \mathbf{w}_2 \; \ldots \; \mathbf{w}_m] \qquad (4)$$

where $\{\mathbf{w}_i | i = 1, 2, \ldots, m\}$ is the set of generalized eigenvectors of S_B and S_W corresponding to the m largest generalized eigenvalues $\{\lambda_i | i = 1, 2, \ldots, m\}$, i.e.,

$$S_B \mathbf{w}_i = \lambda_i S_W \mathbf{w}_i, \quad i = 1, 2, \ldots, m$$

•There are at most c-1 non-zero generalized Eigenvalues, so m ≤ c-1

Generalized eigenvalues and generalized eigenvectors for a pair of symmetric matrices

$$AV = BV\Lambda \text{ and } V^\top A = \Lambda V^\top B$$

where

A is symmetric

B is (symmetric) positive definite

$V = [\mathbf{v}_1 | \cdots | \mathbf{v}_n]$ is not orthogonal

$\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$

Eigenvalue λ_i corresponds to eigenvector $\mathbf{v}_i$.

S_B is A

S_W is B

Fisherfaces

$$W = W_{fld} W_{PCA}$$

$$W_{PCA} = \arg\max_W |W^T S_T W|$$

$$W_{fld} = \arg\max_W \frac{|W^T W_{PCA}^T S_B W_{PCA} W|}{|W^T W_{PCA}^T S_W W_{PCA} W|}$$

- Since S_W is rank N-c, project training set to subspace spanned by first N-c principal components of the training set.
- Apply FLD to N-c dimensional subspace yielding c-1 dimensional feature space.

• Fisher's Linear Discriminant projects away the within-class variation (lighting, expressions) found in training set.

• Fisher's Linear Discriminant preserves the separability of the classes.

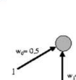
Extra notes about differences between PCA and Fisherfaces

PCA is unsupervised and aims to maximize overall variance, while Fisherfaces (LDA) is supervised and focuses on maximizing the separation between classes. PCA is more concerned with reducing the dimensionality of the feature space efficiently, whereas Fisherfaces is designed to enhance discriminative power.

Neural Networks

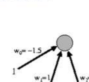
Inverter

input x1	output
0	1
1	0



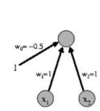
Boolean AND

input x1	input x2	ouput
0	0	0
0	1	0
1	0	0
1	1	1



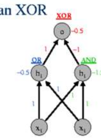
Boolean OR

input x1	input x2	ouput
0	0	0
0	1	1
1	0	1
1	1	1



Boolean XOR

input x1	input x2	ouput
0	0	0
0	1	1
1	0	1
1	1	0



The nodes of multilayered network

If we don't pad x with 1: $y(x;w) = a(w^T x+w_0)$

If we do pad x with 1 for the bias term: $y(x;w) = a(w^T x)$

Activation function: $\tanh(x) = 2/(1+e^{-2x}) – 1$

As x goes from -∞ to ∞, tanh(x) goes from -1 to 1. It has a "sigmoid" or S-like shape. tanh(0) = 0.

Activation function: Rectified Linear Unit (ReLU) g(z) = max(0,z)

Feedforward Network: $y(x) = a_3(a_2(a_1(x; w_1); w_2); w_3)$. Influenced by neuroscience. Our training dictates values to be produced output layer and weights are chosen accordingly. Weights for intermediate or "hidden" layers are learned and not specified directly. Think of network as mapping raw input space x to some transformed feature space Φ(x) where the samples are ideally linearly separable.

Classification Networks: To classify input x into one of c classes, we have c outputs. Output i can be viewed as $p(\omega_i|x)$. That is, the posterior probability of the class, given the input. Recognition decision is $\text{argmax}(p(\omega_i|x))$. If the network were certain about class, one output would be 1 and rest 0. More genearlly, the outputs of c must sum to 1.

Universal Approximation Theorem: A feedforward NN with a linear output and 1+ hidden layers with ReLU, sigmoid, or some other "squashing" activation function can approximate any continuous funciton on closed and bounded subset of R^n. However, optimzation may fail to find parameters needed to represent desired function. Could choose the wrong function due to overfitting. Network required might be so big that it's infeasible. Could fail to train or generalize. Deeper networks are more efficient in that a deep rectified network can represent functions that would require an exponential number of hidden units in a shallow one hidden layer network.

Total Loss: $TL(w) = \text{sum}(L(f(x^i;w)),y^i)$ from i = 1 to n. Types of loss functions below

Distance (regression): L(y,y_hat) = $||y-y\_hat||^p$, usually p = 1 or 2

Softmax Function (classification): softmax(z)_i = exp(z_i) / sum(exp(z_j)) for j in 1 to c

Cross entropy (classification): $\text{sum}(y_i\log(1/y_i\_hat)) - \text{sum}(y_i\log(y_i\_hat))$ where y is a vector with one 1 and the rest 0's while y_hat is a vector with positive floats that sum to 1.

Gradient Descent: Compute gradient $\nabla_x f(x)$ and make step in opposite direction $x_{t+1} = x_t – \eta * \nabla_x f(x_t)$

Backpropagation: iteratively updating the network's weights based on the gradient of the loss function with respect to the weights, calculated through the chain rule of calculus during the forward and backward passes of the network. The goal is to minimize the difference between the predicted output and the actual target values.

Regularization: the goal is to prevent overfitting the training data to improve generalization. L1 (penalty term in loss function proportional to absolute value of weights) and L2 (penalty term is square of weights) regularization encourages sparsity and smaller values in weight matrix respectively.

Dropout: regularization technique that involves randomly "dropping out" a fraction of neurons during training. During each training iteration with SGD, a random subset of neurons is ignored, and their weights are not updated. When done, use the whole model with all weights, except multiply weight by probability of including parent unit (weight scaling inference rule).

Early Stopping: Stop early when the loss of validation set starts to go up. This is to prevent overfitting.

Data augmentation: A way to prevent overfitting. Training samples randomly perturbed to produce more samples.

Deterministic VS Stochastic methods: If performed gradient descent using all training samples, our optimization is *deterministic* (AKA *batch* algorithms). If only used a subset, then called SGD (*mini-batch* algorithms).

SGD: Choose learning rate η_t → Choose stopping critereon → Choose batch size m → Randomly select mini-batch → Forward and backpropagation → Update $\theta_{t+1} = \theta_t – \eta * \nabla_x f(x_t)$ → Repeat 4-6 until stopping criterion satisfied.

SGD with momentum: Compute gradient g. Compute velocity $v_t = \alpha_t * v_{t-1} – \eta_t * g$. Update $\theta_{t+1} = \theta_t – v_t$. α_t starts off small and increases with time (typically). η_t starts large and decreases with time.

Convolutional Neural Networks

Fully conntect layer: Every input unit is connected to every output unit (m * n weights).

Local connection: Connections are spatially located and governed by kernel size. Same weights for each position.

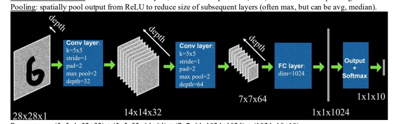
Stride: Allows us to skip input pixels by choosing a stride > 1.

Padding: Given padding size, add the padding size to both sides of the input vector.

Output dimensions: output dimension = (input dim - kernel size + 2 * padding) / stride + 1

ReLU used with CNNs: For convolutional layers, follow linear operation (conv) with non-linear squashing functions.

Pooling: spatially pool output from ReLU to reduce size of subsequent layers (often max, but can be avg, median).



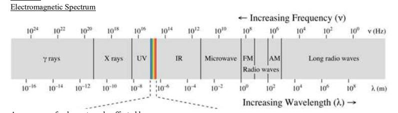
Parameters = (5x5x1x32+32) + (5x5x32x64+64) + (7x7x64x1024+1024) + (1024x10+10)

Famous DNNs: LeNet 5, AlexNet (won ImageNet with 15.3% error rate > 26.2%), VGG16, GoogLeNet, ResNet, PASCAL VOC. ImageNet: Classification task with localization. 20K categories X 1K instances = 14M images.

Domain Transfer (Fine tuning/network surgery): Train DCNN on large problem like ImageNet. Save weights. Chop output layer off and replace with one that fits our problem. Freeze old weight layers and train. Unfreeze and train.

Color

Electromagnetic Spectrum



Appearance of colors: strongly affected by

Spectrum of lighting striking retina: Spectrum of light illuminating a scene influences how colors are perceived. Different light sources emit varying wavelengths, affecting the way colors are reflected and detected by the retina.

Other nearby colors (space): Presence of surrounding colors affect perception of color of a specific object or region.

Adaption to previous views (time): Visual system adapts to the prevailing conditions over time. If exposed to a particular color environment for extended period, eyes can adapt, influencing how we perceive subsequent colors.

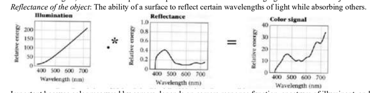
"State of mind": Emotions, mood, and individual differences in perception can impact how we perceive colors.

Talking about colors: *Spectrum*: A positive function over interval 400nm-700nm ("infinite" values needed). *Names*: red, cyan, auburn, etc. (a large, discrete set of color names). *RGB values*: just 3 numbers.

Color Reflectance: Measured color spectrum is a function of the illumination and reflectance.

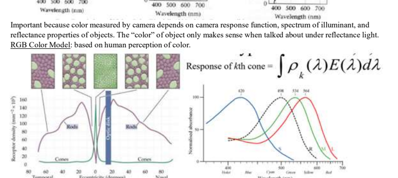
Spectrum of illumination: spectrum of light that illuminates objects. Different light sources, such as natural sunlight or fluorescent lights, have different spectral compositions. Color of illuminating light interacts with object's surface.

Reflectance of the object: The ability of a surface to reflect certain wavelengths of light while absorbing others.



Important because color measured by camera depends on camera response function, spectrum of illuminant, and reflectance properties of objects. The "color" of object only makes sense when talked about under reflectance light.

RGB Color Model: based on human perception of color.



Rods: photoreceptor cells that sensitive to low levels of light, responsible for vision in low-light conditions (scotopic vision). They're not color-sensitive and provide a grayscale image. Concentrated in the peripheral regions of retina.

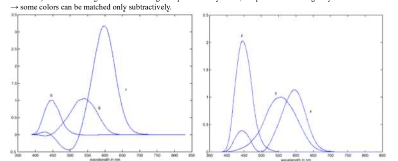
Cones: photoreceptor cells that are responsible for color vision (photopic vision) and function better in bright light. Three types of cones, each sensitive to different ranges of wavelengths corresponding to short (blue), medium (green), and long (red) wavelengths. These cones enable the perception of color by combining responses.

Principle of Trichromacy (color matching experiment): 3 primaries will work for most people if subtactive matching allowed. Exceptional people can match with 1-2 primaries (could be caused by deficiencies). Most people make same matches (there are some anomalous trichromats who use 3 primaries but make different combos to match).

Color matching functions: Choose primaries, say $p_1(\lambda)$, $p_2(\lambda)$, $p_3(\lambda)$. For monochromatic (single wavelength) energy function, what amounts of primaries will match it? That is, for each wavelength λ, how much of p_1, p_2, p_3 needed to match light of that wavelength alone. These are the color matching functions.

RGB: primaries are monochromatic (645.2 nm, 526.3 nm, 444.4 nm). Color matching functions have negative parts → some colors can be matched only subtractively.

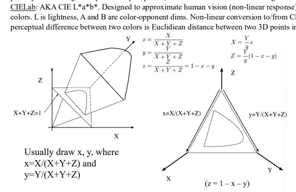
CIEXYZ: Color matching functions are positive everywhere, but primaries are imaginary.



Color spaces: Linear color spaces describe colors as linear combination of primaries. Choice primaries = choice color matching functions = choice color space. Color matching functions/descriptions are all within linear transformations. There are many types: RGB, XYZ, CIExyz, LMS, CMY, YIQ (NTSC), YUV (PAL), YCbCr, SUV, CIELAB. In general, a color represented in one color space can be converted and represented in a second color space.

RGB Color Cube: block of colors for (r,g,b) in range (0-1). Convenient to have upper bound on coefficient of each primary. Primaries given by monitor phosphors (materials on monitor screen that glow when struck by electrons).

CIELab: AKA CIE L*a*b*. Designed to approximate human vision (non-linear response). Includes 100% visible colors. L is lightness, A and B are color-opponent dims. Non-linear conversion to/from CIEXYZ colorspace. Human perceptual difference between two colors is Euclidiean distance between two 3D points in CIELAB space.



$$x = \frac{X}{X+Y+Z} \quad y = \frac{Y}{X+Y+Z} \quad z = \frac{Z}{X+Y+Z} = 1 - x - y$$

$$X = \frac{Y}{y}x \quad Z = \frac{Y}{y}(1 - x - y)$$

Usually draw x, y, where x=X/(X+Y+Z) and y=Y/(X+Y+Z)

(z = 1 – x – y)

CIEXYZ and CIExyz: Encompasses all color sensations the average person can experience. Standard reference (many other colorspaces based on XYZ). Y is luminance. Z is quasi-equal to blue stimulation. X is a linear combination of cone response curves (chosen nonnegative). Plane X+Y+Z=1 contains all possible chromaticities xyz.

Color Specification: Chromaticity Space

Chromaticity coords: (x,y,z) where x+y+z=1. Specified by (x,y) where z=1-x-y. Set of chromaticities RGB and white.

Chromatic Adaption: Estimating appearance of sapmle under a different illuminant – convert between different white points. LMS color space – response of 3 types of cones in human eye (LMS wavelengths). XYZ to LMS – bradford transformation matrix. Chromatic adaption – adaption matrix.

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.8951 & 0.2664 & -0.1614 \\ -0.7502 & 1.7135 & 0.0367 \\ 0.0389 & -0.0685 & 1.0296 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad D = D_1^{-1}D_2 = \begin{bmatrix} L_2/L_1 & 0 & 0 \\ 0 & M_2/M_1 & 0 \\ 0 & 0 & S_2/S_1 \end{bmatrix}$$

Application: Color Transfer: RGB to XYA with white point of standard illuminant E (use chromatic adaption). XYZ to Lab. Map source pixels such that L*a*b* mean and SDs match those of target image. Basically, a technique used to adjust the color characteristics of one image to match those of another.

Nonlinear Encoding: Must conver to linera colorspace first for most color processing. Typical CRT monitors have transfer function of gamma = 2.2. Image and video standards designed to be directly displayed on CRTs. Nonlinear to linear (floating-point) using a lookup table. *Luminance Y and Luma Y'*: Luminance calculated from linear RGB (Y coordinate of XYZ). Luma calculated from nonlinear R'G'B'.

Human Visual System

Why stuudy it in Computer Vision? Computer vision systems mostly operate under same environmental constraints as biological systems. Both exploit same cues for interpreting scenes. Immense diversity in biological vision systems which can serve as models for machine systems. Easier to copy a working system rather than invent from scratch.

Ways to study human vision:

Physiological: involve examining the anatomical and functional aspects of the visual system at physiological level.

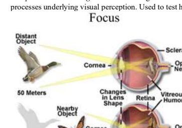
Phenomenological/Psychophysical: focus on understanding the subjective experiences and perceptions of individuals in response to visual stimuli.

Cellular recordings: measuring the electrical activity of individual neurons or groups of neurons in the visual system.

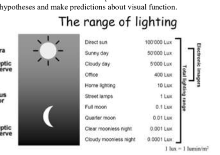
Functional MRI: neuroimaging technique that measures changes in blood flow associated with neural activity. Participants undergo fMRI scans while engaging in visual tasks. Resulting images reveal brain activation patterns.

Computational modeling: involves creating mathematical models or simulations to represent and understand the processes underlying visual perception. Used to test hypotheses and make predictions about visual function.

Focus



The range of lighting



Other eyes:

Trilobite Visual System: Most ancient known visual system. Compound eye with single crystal for each lens.

Scallop eyes: Hundreds of primitive eyes, mirror in back. Changes in light and motion and very rough images are registered on the retinas of the mollusk.

Stomatopod eyes: Dumb bell shaped, compound eyes. Stereo vision with just one eye. Each eye is up on a stalk, with a wide range of motion. Stomatopods have up to 16 visual pigments. Stomatopods can see ultra-violet and infra-red light. Some can see polarized light.

Cues: pieces of information that visual system uses to interpret the environment and make sense of the surrounding world. In human vision, cues help perceive depth, distance, size, shape, motion, and other characteristics of objects.

Null Space and SVD

(Right) null space

The (right) null space of a matrix $A \in \mathbb{R}^{m\times n}$ is the matrix $X = \text{null}(A)$ such that

$$AX = 0$$

where $X \in \mathbb{R}^{n\times(n-r)}$ and $r = \text{rank}(A) \leq \min(m, n)$.

Left null space

The left null space of a matrix $A \in \mathbb{R}^{m\times n}$ is the matrix Y such that

$$YA = 0$$

where $Y \in \mathbb{R}^{(m-r)\times m}$ and $r = \text{rank}(A) \leq \min(m, n)$. The left null space may be calculated using the (right) null space as $Y = (\text{null}(A^\top))^\top$.

Computation of the right and left null space using SVD

The singular value decomposition (SVD) of a matrix $A \in \mathbb{R}^{m\times n}$ may be written as

$$A = U\Sigma V^\top$$

where the orthogonal matrix $U \in \mathbb{R}^{m\times m}$, the diagonal matrix $\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_{\min(m,n)}) \in \mathbb{R}^{m\times n}$, where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(m,n)} \geq 0$, and the orthogonal matrix $V \in \mathbb{R}^{n\times n}$. σ_i is the i-th singular value of A, and the i-th column of U and i-th column of V are the corresponding left singular vector and right singular vector, respectively, of A. The rank r of A is the number of nonzero singular values. The (right) null space of A is the columns of V corresponding to singular values equal to zero. The left null space of A is the rows of $U^\top$ corresponding to singular values equal to zero (or the columns of U corresponding to singular values equal to zero, transposed).