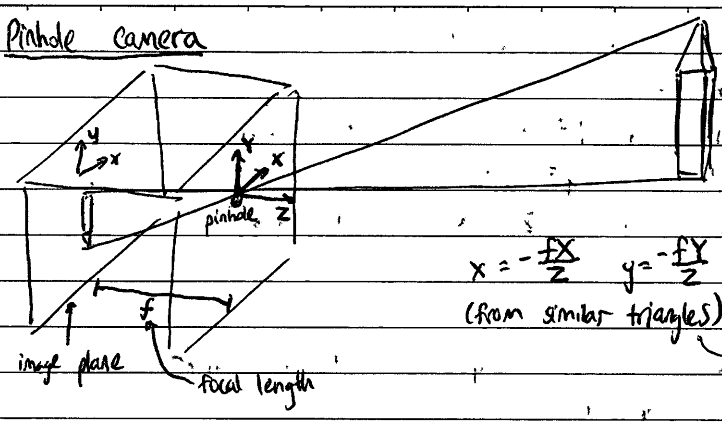
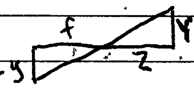
CS C280, Spring 2018

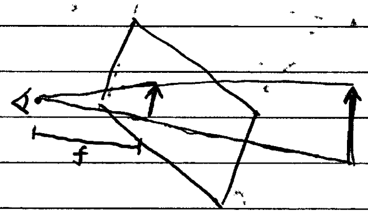
Pinhole camera



$x = -\frac{fX}{Z}$ $y = -\frac{fY}{Z}$
(from similar triangles)

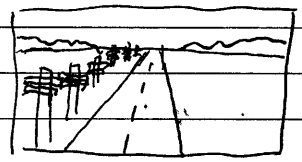


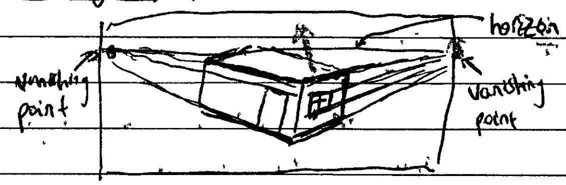
but the image is flipped! so we'll use perspective projection instead:



$x = \frac{fX}{Z}$, $y = \frac{fY}{Z}$

interesting: parallel lines converge, and each family of parallel lines converges to its own vanishing point (all of which lie on the horizon)





why do parallel lines converge?

a line is given by $\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \underbrace{\begin{bmatrix} A_X \\ A_Y \\ A_Z \end{bmatrix}}_{\text{initial point}} + \lambda \underbrace{\begin{bmatrix} D_X \\ D_Y \\ D_Z \end{bmatrix}}_{\text{direction}}$

consider the projection: $x = \frac{fX}{Z} = \frac{f(A_x + \lambda D_x)}{A_Z + \lambda D_Z}$ $y = \ldots$

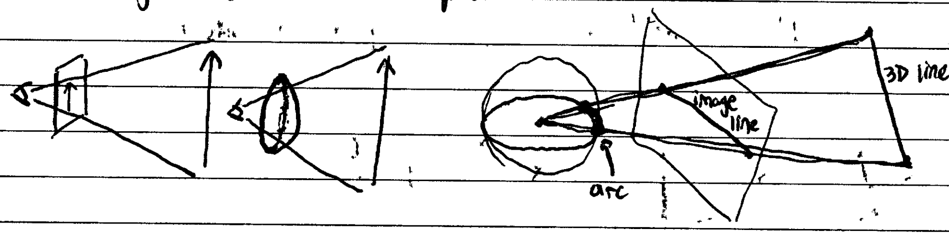
now if $\lambda \to \infty$ $x \to \frac{f \lambda D_x}{\lambda D_Z} = \frac{f D_x}{D_Z}$ $y \to \frac{f D_y}{D_Z}$

these don't depend on $\begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix}$ (~~these are~~ this is the vanishing point)

what about vertical lines? they don't vanish, because $D_Z = 0$

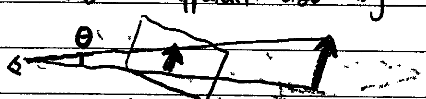
why are nearer objects on the ground plane lower in the image?
let the camera height be h
then the ground plane has $Y=-h$ and therefore an object on the ground has $y=\frac{-fh}{Z}$
so larger Z means larger y: $Z \to \infty$ leads to $y \to 0$.
by extension, nearer objects look bigger

perspective projection does not assume the imaging surface to be planar;
it could just as well be spherical:



the retina is in fact closer to spherical than planar.

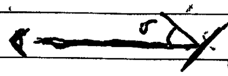
we can measure apparent size by visual angle:



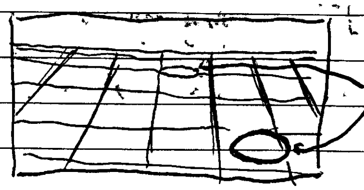
by combining this with some notion of the height of the object, you can determine its distance

two main effects of perspective projection:

1. distance: farther objects have smaller apparent sizes (with scaling factor $1/Z$)
2. foreshortening: objects slanted w.r.t. the line of sight project to smaller apparent sizes (with scaling factor $\cos \sigma$)



σ is angle between line of sight and surface normal



these disks are actually circles, but appear as ellipses (farther → flatter)

No.

DATE

when an object is far away relative to the depth variation in it, we can approximate perspective using orthographic projection

this changes the perspective scaling factor f/Z to a constant, $s = f/Z_0$

the projection equations are then $x = sX$ and $y = sY$ (simpler!)

there are no vanishing points in orthographic projections!

pose: how an object is positioned and oriented relative to the observer, as defined by 6 numbers, 3 for translation and 3 for rotation

shape: the coordinates of points on an object relative to a coordinate frame on the object (these are rotation- and translation-invariant)

rigid ~~object~~ body: distances between points on the object remain constant

isometry: distance-preserving transformation, Ψ where

$$\|a-b\| = \|\Psi(a) - \Psi(b)\|$$

e.g. ~~transformations~~ translations, $\Psi(a) = a+t$, because

$$\|\Psi(a)-\Psi(b)\| = \|a+t-(b+t)\| = \|a-b\|$$

orthogonal transformation: linear transformation (i.e., $\Psi(a) = Aa$ for some ~~[illegible]~~ matrix A) that preserves inner products:

$$a \cdot b = \Psi(a) \cdot \Psi(b)$$

includes rotations and reflections

all orthogonal transformations are isometries

for orthogonal transformations, A must be orthogonal: $A^TA = AA^T = I$

A is square and all rows/columns are orthonormal

note that $\det(A) = \pm 1$

theorem: any isometry can be expressed as an orthogonal transformation followed by a translation

in 2D, there are only two kinds of orthogonal matrices:

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix}$$

rotation, $\det = +1$ reflection, $\det = -1$ (around line w/ angle $\theta/2$)

in 3D, given rotation angle ϕ and direction $\hat{s}$ (unit vector) we can find the rotation matrix using Roderigues's Formula:

$$R = e^{\phi S} = I + (\sin\phi)S + (1-\cos\phi)S^2 \quad \text{where} \quad S = \begin{bmatrix} 0 & -\hat{s}_3 & \hat{s}_2 \\ \hat{s}_3 & 0 & -\hat{s}_1 \\ -\hat{s}_2 & \hat{s}_1 & 0 \end{bmatrix}$$

No.

DATE

affine transformation: $\varphi(a) = Aa + t$ where A is non-singular ($\det A \neq 0$).
this is a superset of orthogonal transformations and isometries

let's count degrees of freedom:

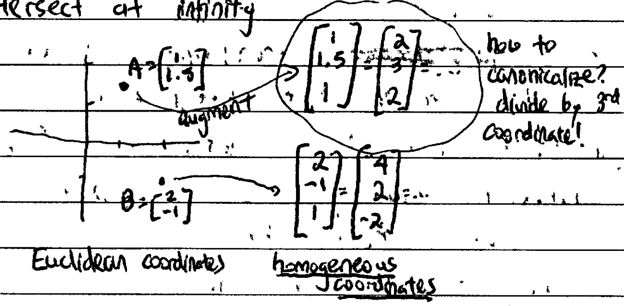
in 2D, isometries have 3 free parameters (1 rotation, 2 translation)
affine transformations have 6 (4 in A, 2 in t)

in 3D, isometries have 6 free parameters (3 rotation, 3 translation)
affine transformations have 12 (9 in A, 3 in t)

affine transformations preserve straight lines and pairs of parallel lines
e.g. scaling or shearing

projective space: Euclidean space modified to add rules true of perspective
e.g., "parallel lines intersect at infinity"

from $\mathbb{R}^3$ (excluding $\vec{0}$) we can construct P^2 by saying $\vec{p}$ and $\vec{p}^*$ are equivalent if $\vec{p} = \lambda \vec{p}^*$ for some $\lambda \neq 0$



we canonicalize a triplet of homogeneous coordinates by dividing by the 3rd coordinate
but what if it's zero? this corresponds to a "point at infinity"

consider the projective line (P^1)

any finite point x can be represented as

$$\begin{bmatrix} x \\ 1 \end{bmatrix} \text{ or } \begin{bmatrix} 2x \\ 2 \end{bmatrix} \text{ or } \begin{bmatrix} 6.3x \\ 6.3 \end{bmatrix} \text{ or} \ldots$$

any infinite point can be represented as

$$\begin{bmatrix} x \\ 0 \end{bmatrix}$$

(there is only one such point) → zero degrees of freedom

consider the projective plane (P^2):

any finite point can be represented as

$$\begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix}$$

any infinite point can be represented as

$$\begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

(different $x:y$ ratios give different points, so there is a line at infinity) → one degree of freedom how do we represent a Euclidean line in homogeneous coordinates?

$a_1 x + a_2 y + a_3 = 0$

$\rightarrow a_1\left(\frac{x}{z}\right) + a_2\left(\frac{y}{z}\right) + a_3 = 0$

$\rightarrow a_1 x + a_2 y + a_3 z = 0$

when does a point lie on the line? if $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$

$\vec{a} \cdot \vec{x} = 0$ $\quad \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$

any two lines intersect!

$\vec{a} \cdot \vec{x} = 0$ and $\vec{b} \cdot \vec{x} = 0$ intersect if there exists $\vec{x}$ perpendicular to both $\vec{a}$ and $\vec{b}$

but we can easily construct such a point: $\vec{a} \times \vec{b}$

example: $x=1$ and $y=1$ intersect at $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \mapsto (1, 1)$

example: $x=1$ and $x=2$ intersect at $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} -1 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ (point at infinity)

projective transformation: linear transformation ($\varphi(a) = Aa$ for non-singular A) in homogenous coordinates, e.g., a 3×3 invertible matrix defines a 2D projective transformation

this is a superset of affine transformations

e.g. in 2D:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \quad w' = 1$$

(linear transformation in P^2) (affine transformation in $\mathbb{R}^2$) (the $\begin{bmatrix} x \\ y \end{bmatrix}$ in P^2 match the $\begin{bmatrix} x \\ y \end{bmatrix}$ in $\mathbb{R}^2$)

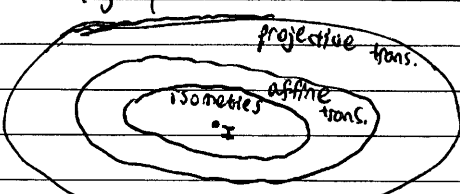
we can also represent perspective projection:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z/f \end{bmatrix} \Rightarrow \begin{bmatrix} fx/z \\ fy/z \\ 1 \end{bmatrix}$$

(in P^3)

a projective transformation in P^2 is a 3×3 matrix, but there are only 8 independent parameters, since A and kA are the same transformation

the final big picture:



No.

DATE . .

optical flow: movement of a point on the image plane caused by the movement of a point in the world relative to the camera.
(what is important is relative motion: it does not matter if the camera moves or real objects move)

goal: relate the optical flow field to scene depth $Z(x,y)$ and the camera motion (given by t, translation, and ω, rotation)

$\frac{dX}{dt}$ for point X in scene: $\dot{X} = -t - \omega \times X$

taking into account projection (assume $f=1$):

$$\dot{x} = \frac{\dot{X}Z - \dot{Z}X}{Z^2}, \quad \dot{y} = \frac{\dot{Y}Z - \dot{Z}Y}{Z^2}$$

Substituting and plugging in yields:

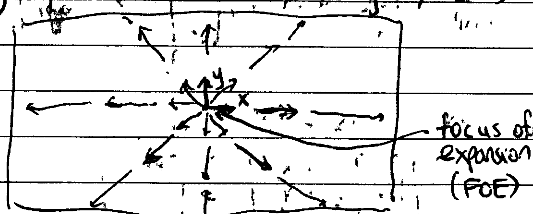
$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{1}{Z}\begin{bmatrix} -1 & 0 & x \\ 0 & -1 & y \end{bmatrix}\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} xy & -(1+x^2) & y \\ 1+y^2 & -xy & -x \end{bmatrix}\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

also written $[u(x,y) \;\; v(x,y)]^T$

example case: translation along optical axis ($\omega=0$, $t_x=t_y=0$, $t_z \neq 0$)

$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \frac{t_z}{Z}\begin{bmatrix} x \\ y \end{bmatrix}$$

optical flow vector is scalar multiple of position vector



for general translation ($\omega=0$):

$u(x,y) = \frac{-t_x + x t_z}{Z}$ the FOE is $[t_x/t_z \;\; t_y/t_z]^T$

$v(x,y) = \frac{-t_y + y t_z}{Z}$ (the point where $u(x,y) = v(x,y) = 0$)

if we changed the origin to the FOE, then the optical flow field would be similar to the optical axis-only translation case, so even in the general case, the optical flow vectors point outward from the FOE

in rotation-only case, we can determine ω from the optical flow field

No.

DATE

irradiance: power per unit area (W/m²)

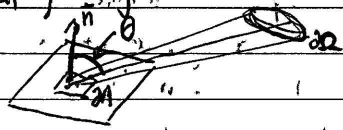
radiance: radiant power emitted/reflected by a surface per unit solid angle per unit area ($\frac{W}{sr \cdot m^2}$), as a directional quantity

$$L = \frac{\partial^2 \Phi}{\partial \Omega \, \partial A \cos\theta}$$

← power

effective receiving area

solid angle (3D analog of 2D angles in radians; measured in steradians)



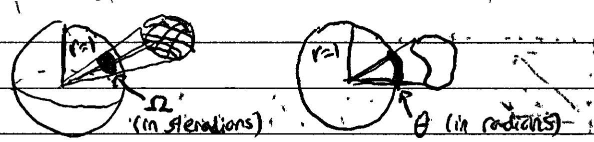
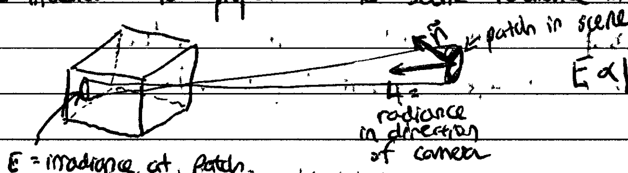


image irradiance is proportional to scene radiance in direction of camera (pinhole or lens)



$E \propto L$

what causes outgoing radiance from scene patch? 1) incoming radiance from light source, 2) angle between patch normal and incoming light, and 3) reflectance properties of the patch

specular surface: outgoing radiance direction obeys $\theta_{incidence} = \theta_{radiance}$, as in a mirror

Lambertian surface: outgoing radiance same in all directions, as in a "matte" surface

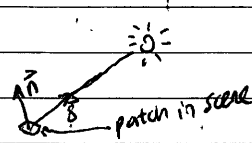
these are idealized surfaces

Lambertian model describes radiance:

$L = \rho\lambda(\hat{n} \cdot \hat{s})$ ← corresponds to foreshortening

albedo depends on irradiance of light source

(roughly, measures the light absorptivity of a surface, from 0 [absorbs everything] to 1 [reflects everything])



this holds for every particular wavelength, and variations in amounts of light of different wavelengths gives rise to the perception of color

edges in images are curves across which brightness changes a lot changing any term in the Lambertian model can cause an edge in the image

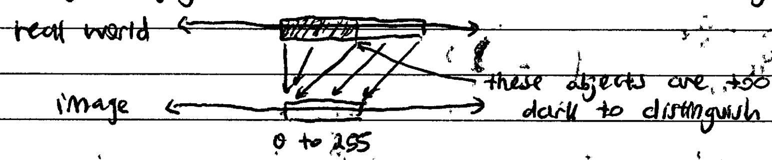
<u>image</u> = 2D array of intensities (or perhaps multiple intensities for color)

$f(x, y)$ = reflectance at (x, y) × illumination at (x, y)

intensity at (x, y) — in $[0, 1]$ — in $[0, \infty)$

the real world has high dynamic range of intensities

during imaging this is mapped to $[0, 255]$ (typically lossily)



pixel values do not correspond to true light intensities (due to various nonlinear stages in image acquisition pipeline)

<u>point processing</u>: transformation of an image independent of location

$g(x, y) = T(f(x, y))$

examples: negative, contrast stretching (e.g., based on histograms), gain-and-bias ($g = af + b$)

as with audio, images are lossy due to sampling / quantization

<u>aliasing</u>: different signals become indistinguishable due to sampling

<u>cross-correlation</u>: shift a kernel around an image, taking a dot product at each position

$$\underset{\text{output}}{G[i,j]} = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \underset{(2k+1,\ 2k+1)\text{ kernel}}{H[u,v]}\ \underset{\text{image}}{F[i+u, j+v]}$$

$G = H \otimes F$

this is a linear filter: filter$(a+b)$ = filter(a) + filter(b)

also, it is shift-invariant: the same operation is used in every part of the image

example: <u>box filter</u> (applies blur)

$$H = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

No.

DATE . .

alternative: Gaussian kernel (like box filter but points farther from kernel center have less weight per the Gaussian PDF).

parameterized by σ, ~~kernel width~~ blur amount (and kernel size)

convolution: cross-correlation where the filter is flipped horizontally and vertically first

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

$$G = H \star F$$

convolution has nice properties:

- commutative: $a * b = b * a$ (no distinction between filter and signal)
- associative: $a * (b * c) = (a * b) * c$
- distributes over addition: $a * (b + c) = a * b + a * c$

associativity is useful:

$$(((a * b_1) * b_2) * b_3) = a * \underbrace{(b_1 * b_2 * b_3)}$$

image filters

filters are smaller, so this saves time

a Gaussian is a low-pass filter (~~removes~~ attenuates high-frequency components)

convolving two Gaussians results in another Gaussian

convolution theorem: convolution in spatial domain is equivalent to multiplication in frequency domain

$$F[g * h] = F[g] \cdot F[h]$$

$$F^{-1}[g \cdot h] = F^{-1}[g] * F^{-1}[h]$$

where $\cdot$ is pointwise multiplication

Fourier transforms are lossless (they're like a change of basis)

the Fourier transform of a Gaussian is Gaussian

common use of filtering: apply Gaussian before subsampling (avoids aliasing)

to do this quickly, use a pyramid: repeatedly blur and subsample by factor of 2 (instead of using a huge Gaussian blur followed by a huge downsample)

this kind of multi-scale processing is a recurring theme

No.

DATE

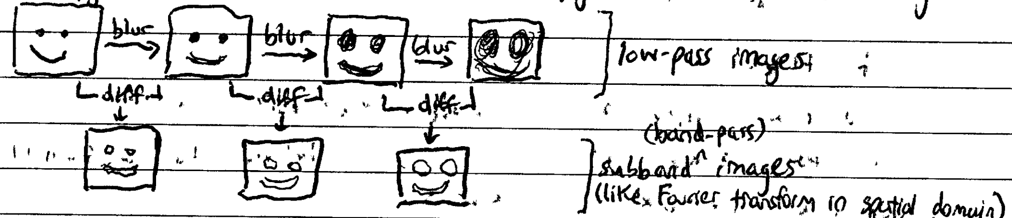
<u>sharpening</u> is done by adding in high frequencies:

sharpened = original + $\alpha \cdot$ (original − blurred)

humans have limited contrast sensitivity

e.g. the sky in an image may look "blue" while actually being comprised of many different colors

<u>Laplacian pyramid</u> formed from Gaussian pyramid forms subband images



(example does not subsample, so technically this is a "stack", not a pyramid)

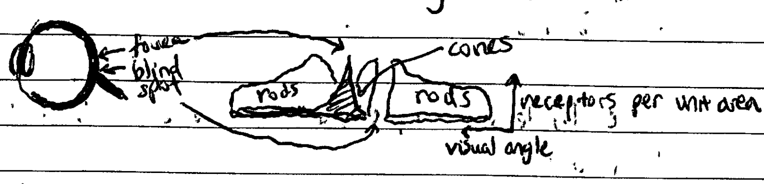
subband images (and the final leftover low-pass image) can be collapsed to recover the original image

this can be done with modifications to achieve various effects

e.g., blending orange and apple smoothly

<u>Saccadic eye movements</u>: movements of the eye scanning a scene (receptive field is blurry at the periphery)

the retina of the eye contains sensors called <u>rods</u> (grayscale vision, highly sensitive) and <u>cones</u> (color vision, less sensitive), which are not distributed uniformly:



thus, visual accuity is non-uniform (periphal vision is worse)

Da Vinci's Mona Lisa's smile is mysterious because the high-frequency details are those of a stern half-smile but the coarse components reflect a full smile

as a result, she appears to smile in your peripheral vision but not when looking directly at her

No.

DATE . .

there are three kinds of cones (RGB traditionally, now S, M, and L)

green cones are more common, so we are more sensitive to green light

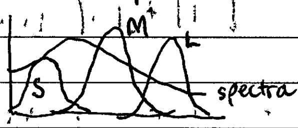
why do we see the range of wavelengths that we do?

evolution: our sun produces light in that range

objects have colors because they absorb/reflect light of different wavelengths

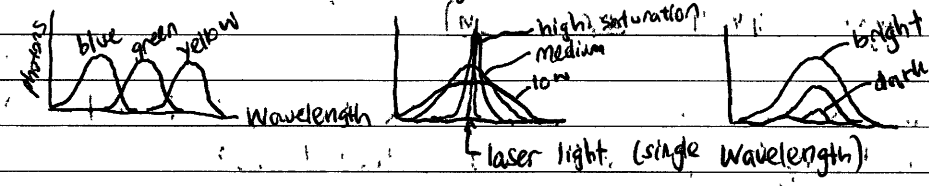
note that color is a psychological phenomenon; only wavelengths are physical

Metamers: spectra that appear to be the same color, due to the lossiness of RGB/SML representation (trichromacy)



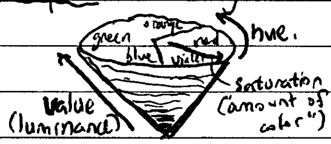
find S value: pointwise-multiply S curve with spectra, then compute integral

supposing spectra are normally distributed, mean corresponds to hue, variance to saturation, and area to brightness



RGB color space: easy for machines, less intuitive for people

HSV color space: used by artists



Lab color space: perceptually uniform (distances between points reflect subjective distances between colors)

color constancy: perceived colors remain invariant under varying illumination conditions

this is an example of why ~~vision~~ isn't analogous to a photometer

White-balancing: force the brightest object to be white and the average color to be gray

corrects for illumination of scene, which may cause colors to look different in photos than in real life

ultimately perception of color is underdetermined by the physics of light and surface reflectance

No.

DATE

finding an edge is mathematically related to taking a derivative:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon} \approx \underbrace{\frac{f(x+1,y)-f(x,y)}{1}}$$

finite-difference approximation — corresponds to convolution filter $\boxed{-1} \boxed{+1}$

Other edge-detection filters exist:

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Prewitt (detects an "edgelet" of width 3)

Sobel (middle pixels are more important since response is assigned to middle pixel) — $[1\ 1] * [1\ 1] = [1\ 2\ 1]$

Roberts (finite-difference method applied between pixels)

gradient of an image shows direction of most rapid intensity increase

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

direction: $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$

Magnitude: $\|\nabla f\|$ (edge strength)

what if the original signal has noise, so $\frac{\partial f}{\partial x}$ is noisy (hard to find edge)?

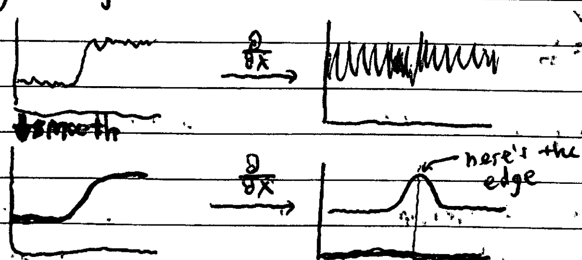
first smooth signal with Gaussian, then derive:

$$\frac{\partial\, f * g}{\partial x} = f * \frac{\partial g}{\partial x}$$

f: image, g: Gaussian

(differentiation is convolution and convolution is associative/commutative)

smoothing averages local variation to zeroes:



but smoothing also blurs edges, so there is a trade-off between smoothing and good edge localization

Canny edge detector: multi-stage edge detection algorithm that resolves blurring issue

Canny edge detector steps:

1) smooth then compute gradient (as before)
2) apply <u>non-maximum suppression</u> to thin edges: set gradient to zero if not local maximum along gradient direction (may require bilinear interpolation to find values of other points along gradient direction)
3) apply high and low thresholds to find strong and weak edges
4) <u>hysteresis</u>: eliminate weak edges not connected to strong edges

how do we find a particular object in an image using a template?

1) filter with zero-mean template:

$$h[m,n] = \sum_{k,l} (g[k,l] - \bar{g})(f[m+k, n+l])$$

image; template; mean of template

this leads to false detections, but mostly works: the zero-mean template acts like a derivative filter (since its sum is zero, constant patches will have value zero and patches similar to the template will have high value)

2) <u>SSD</u>:

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$

(result will be low-valued in similar regions)

this works but is sensitive to scaling by a constant

3) <u>normalized cross-correlation</u>: not sensitive to scaling, but is slower

types of recognition:

<u>instance recognition</u>: find a particular object (template matching works okay)
<u>category recognition</u>: find a type of object, e.g., chairs
requires focusing on invariant attributes across the category

<u>texture</u> comes from repeating patterns in "stuff"

distinct modes of vision:

<u>preattentive vision</u>: parallel, instantaneous vision covering a large visual field
<u>attentive vision</u>: serial, linear search limited to small aperture.

A A A   A A A
A ∀ A   A **A** A
A A A   A A A

changes in orientation/size are discriminated very quickly during the preattentive phase

texture arises from patterns not preattentively discriminable

No.

DATE . .

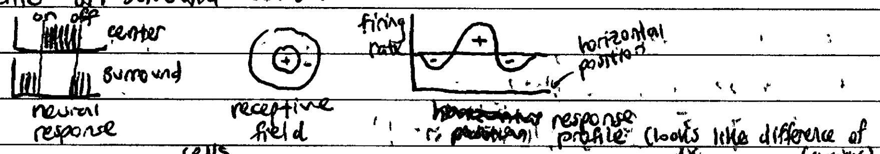
Julesz conjecture: textures with the same first-order statistics (density) and second-order statistics (relationships between pairs of points) cannot be spontaneously discriminated.

This conjecture isn't fully true, but is useful

Discovered by psychophysicist Béla Julesz by presenting subjects with stimuli (drawings)

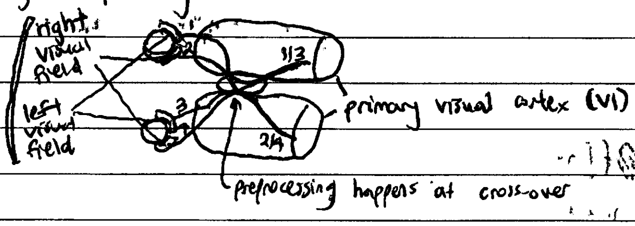
Receptive field: area in visual field "seen" by a given cell

on-center off-surround cells:

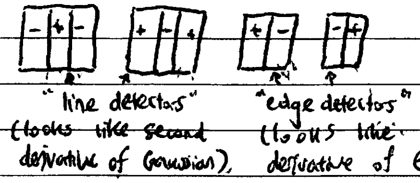


receptive field grows for cells later and later in the vision process

anatomy of pathway to visual cortex:



Hubel and Wiesel discovered in 1959 (by chance) that certain cells in V1 (simple cells) responded to oriented lines



there are also complex cells (like simple cells but with spatial invariance), which inspired max pooling in CNNs

hypercolumn: all cells corresponding to a receptive field, which detect various patterns at different scales, like a filter bank

Hubel and Wiesel theorized that the vision process is a hierarchy of features detectors, from simple to complex/specialized

recent psychophysics result: basic object detection ("is there an animal?") is very fast (150ms), requiring only preattentive vision (perhaps only texture!)

how can we capture texture, e.g., for recognition?

1) use filters and compute the mean for each response (then feed these features to a classifier for classification)

2) use filters but now compute histograms of filter responses (one per filter)

3) use filters but histograms of joint response (so that filters can "talk to each other")

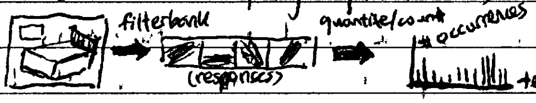
#3 allows for semantic filters corresponding to visual words (by analogy to bag-of-words models), high-level features in an image (e.g., an eye or ear)

how to create dictionary of visual words?

extract patches from images, then apply K-means clustering

codewords created in this manner are called textons

image classification using textons (early 2000s): for test image, gather patches, quantize each to nearest codeword (after applying filters), create histogram of counts of textons, then compare histogram to known/labeled histograms using χ^2-test (similar to computing squared error)



this tends to work well (despite being very simple, using only low-level statistics of image patches)

Kinds of histograms:

joint histogram (requires lots of data, but is otherwise better)

marginal histogram (doesn't capture correlations)

Many featurization techniques use histograms (or other aggregations) of low-level features: SIFT, GIST, shape context, spatial pyramid matching, etc.

alternative to using a pre-chosen set of filters: learn filters during training

Olshausen and Field proposed a technique for this in the late 1990s

their loss function contained a reconstruction error term and a sparsity penalty

this paper was exciting because the learned filters looked similar to the ones in V1 discovered by Hubel and Wiesel

this is how the world looked when neural networks started becoming popular

No.

DATE . .

<u>neocognitron</u> (Fukushima): 1980 neural network ~~model~~ architecture modeled after Hubel and Wiesel's ideas of hierarchy for unsupervised featurization

this model did not have backprop but did have max pooling ("complex cells") and ReLU non-linearities

<u>convolutional neural networks</u>: introduced in 1998 by LeCun for supervised classification (MNIST)

trained via backprop

<u>LeNet-5</u> (LeCun et al., 1998): conv-pool-conv-pool-conv-FC architecture for MNIST

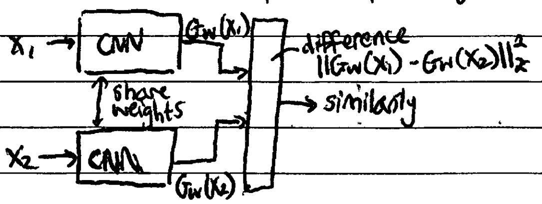
<u>AlexNet</u> (Krizhevsky et al., 2012): ReLU, norm, dropout, augmentation, ...

<u>ResNet</u> (He et al., 2015): 152 layers, requiring 2-3 weeks of training on 8 GPUs
includes residual connections (earlier inputs are available to later layers)

activations at various CNN layers can be used as featurizations

transfer learning with CNNs is easy: retrain some of the later layers on a pre-trained network

<u>Siamese network</u>: takes separate input images and outputs similarity



trained via <u>contrastive loss</u>:

$L_p(x_q, x_p) = \|f(x_q) - f(x_p)\|_2^2$ if x_q, x_p similar

$L_n(x_q, x_n) = \max(0, m^2 - \|f(x_q) - f(x_n)\|_2^2)$ if x_q, x_n dissimilar

negative positive margin

$$L(\theta) = \sum_{(x_q, x_p)} L_p(x_q, x_p) + \sum_{(x_q, x_n)} L_n(x_q, x_n)$$

penalty for similar images far away

penalty for dissimilar images close together

can be used to find products via image search

No.

DATE . .

also possible: multi-modal architectures

image → [CNN] → [ ] → matching

text → [Word2vec] → [ ] →

relative position task: given patch from image and another test patch, where is the test patch located relative to the other patch?

can also perform many tasks at once:

image → [CNN] → [CNN] → ... → [CNN] → [FC] → [FC] → task 1
→ [FC] → task 2
→ [FC] → task N

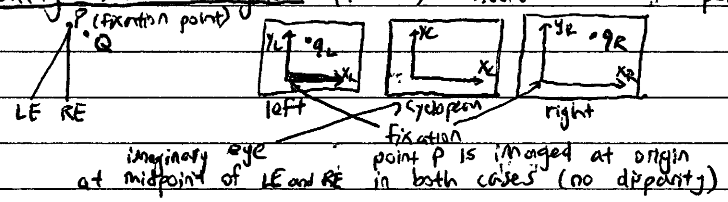
in theory, doing many tasks at once should benefit all tasks

in general, you can put neural network components in any DAG

LeCun: "Deep Learning est mort. Vive Differentiable Programming!"

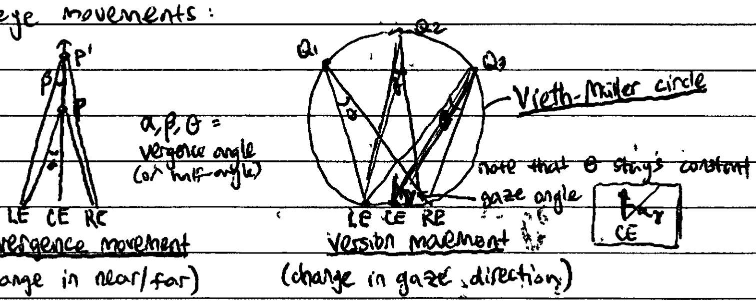
binocular vision gives rise to depth perception because of disparity in images

fixating binocular system (two eyes focused on same point):

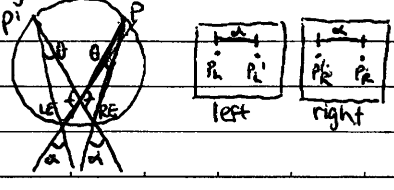


we will measure world coordinates relative to the cyclopean eye

basic eye movements:



disparity is zero on the Vieth-Müller circle:

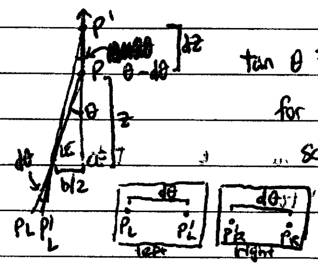


No.

DATE

how does disparity relate to depth?

baseline (b) is distance between eyes



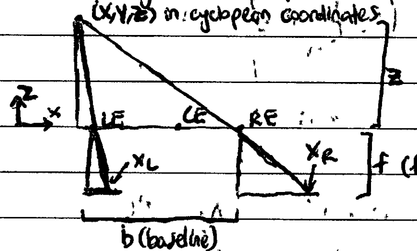
$\tan\theta = \frac{b/2}{z}$ $\tan(\theta - d\theta) = \frac{b/2}{z+dz}$

for small θ, $\tan\theta \approx \theta$

so $d\theta = \frac{b/2}{z} - \frac{b/2}{z+dz} \approx \frac{b\,dz}{2z^2}$

so disparity $= \underbrace{2\,d\theta}_{d\theta \text{ in left, } d\theta \text{ in right}} \approx \frac{b\,dz}{z^2}$

now consider a system with parallel optical axes (fixation at infinity):



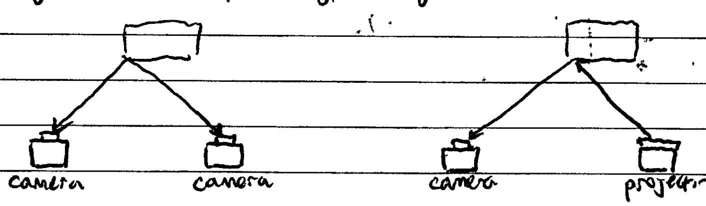
from similar triangles:

$\frac{x_L}{f} = \frac{X - b/2}{Z}$

$\frac{x_R}{f} = \frac{X + b/2}{Z}$

disparity is $x_R - x_L = \frac{bf}{Z}$ (notice $Z \to \infty$ leads to no disparity)

two ways to sense depth using triangulation:



passive stereopsis

active stereopsis (e.g., Kinect)

both have the same geometry, so the disparity formula is the same

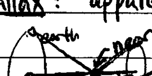
for parallel optical axes, what is depth error in terms of disparity?

$\overset{\text{depth}}{Z} = \overset{\text{constant}}{c}/d \leftarrow \text{disparity}$ thus $\frac{\delta Z}{\delta d} = -c/d^2 = -Z^2/c$

so $|\delta Z| \propto Z^2$

this has been empirically confirmed for the Kinect.

parallax: apparent displacement of objects due to movement of the observer.



parallax angle

far star (appears to move in ellipse relative to near star as Earth moves)

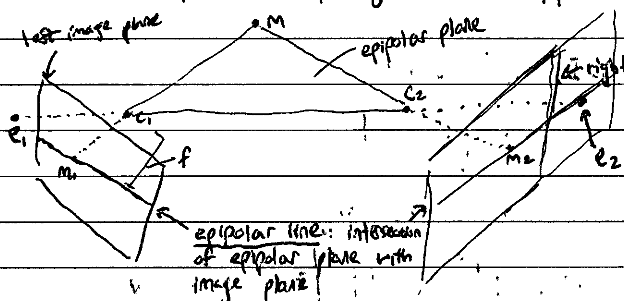
No.

DATE

general case: two cameras related by rotation R, translation $\mathbf{t}$ (both unknown)

<u>epipolar plane</u>: plane formed by target point (in world), M, and the centers of projection, C_1 and C_2

different world points have (possibly) different epipolar planes, but all include the line $\overline{C_1C_2}$



epipoles e_1 and e_2 are the same for all epipolar planes

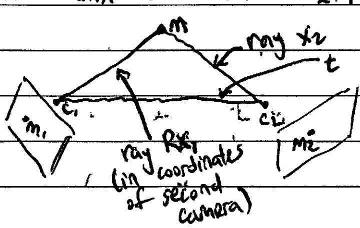
Structure from motion problem: setup above with n world points (X_i, Y_i, Z_i) and n projected points for each camera

approach:

1) find n corresponding points in the two views
2) estimate "essential matrix" $E = \hat{T}R$ from the correspondences (where $\hat{T}$ is the skew-symmetric matrix corresponding to translation t)
3) extract R and t (via factorization)
4) recover depth by triangulation

this is called <u>bundle adjustment</u>, a nonlinear least squares problem that minimizes reprojection error

essential matrix constraint: $x_2^T \hat{T} R x_1 = 0$, where x_1, x_2 are m_1, m_2 in homogenous coordinates



Rx_1, x_2, t are coplanar

$\rightarrow x_2 \cdot (t \times Rx_1) = 0$

$\rightarrow x_2^T \underbrace{\hat{T}R}_{E} x_1 = 0$

$E = \hat{T}R$ has 8 degrees of freedom (since $\lambda E = E$, as with projective matrices in homogenous coordinates), so we can solve for E using 8 points (x_1, x_2);

this is the <u>Longuet-Higgins 8-point algorithm</u> (for two views)

bundle adjustment is more specifically reconstruction from many cameras, using Longuet-Higgins for initialization

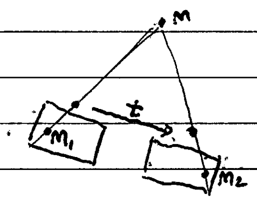
No.

DATE . .

stereo matching: finding corresponding points (m_1, m_2) in images from two cameras

m_2 must lie on the epipolar line (defined by the epipolar plane) on the right image

simple case: parallel images



$R = I, \; t = [T, 0, 0]$

$\rightarrow E = \hat{T}R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix}$

constraint: $\underbrace{m_2^T E m_1 = 0}$

$[u' \; v' \; 1] \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0 \rightarrow Tv' = Tv$

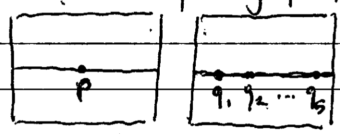
so $v' = v$ (y-components of corresponding points are equal)

so the epipolar lines are horizontal scan lines

stereo image rectification: reprojecting image planes to match above situation

requires two homographies (3×3 transforms) computed based on $R, \hat{T}$

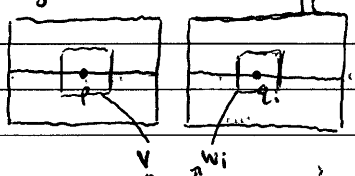
how to find corresponding points if epipolar lines are horizontal?



(i.e., assuming we have performed rectification)

photoconsistency assumption: corresponding points have similar brightnesses in the two photos (based on Lambertian model)

this gives us a naive approach:



two similarity functions (matching costs):

SSD: $\|v - w_i\|_2^2$ (smaller is more similar)

NCC: $\frac{w_i \cdot v}{\|v\|\|w_i\|}$ (larger is more similar)

v, w_i — vectors representing pixels in windows (window size controls robustness)

once corresponding points are known, we can compute the depth

$Z = \frac{bf}{x - x'}$] - disparity

when does the naive solution fail?

textureless surfaces, e.g., a blank wall (w_i is the same everywhere)

occlusions, repetition

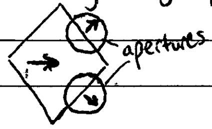
non-Lambertian surfaces, e.g., a mirror

picking a window size is also hard

smaller means more noise but more detail

larger means smoother but less detail

__aperture problem__: sometimes, the true motion of an object cannot be inferred (is ambiguous) when viewing only part of the object



how do we calculate optical flow, $(u, v) = \left(\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t}\right)$?

brightness constancy assumption: $I(x_1, y_1, t_1) = I(x_2, y_2, t_2)$ for corresponding points

differenting, we have:

$$dI = I_x\,dx + I_y\,dy + I_t\,dt = 0 \quad \text{where } I_x = \frac{\partial I}{\partial x}, \text{ etc.}$$

divide by dt:

$$I_x \frac{dx}{dt} + I_y \frac{dy}{dt} + I_t = 0$$

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

$\nabla I = [I_x \; I_y]^T$ is known (from images), as is I_t

so our equation is

$$\nabla I \cdot \tilde{u} = -I_t \quad \text{where } \tilde{u} = [u \;\; v]^T$$

so the length of $\tilde{u}$ in the direction of ∇I is known, but not the length of $\tilde{u}$ in the direction perpendicular to ∇I.

this is how the aperture problem arises! (the brain assumes that the component of $\tilde{u}$ perpendicular to ∇I is zero)

we resolve this by using multiple nearby points and assuming (u, v) are the same across these points (local constancy of optical flow)

$$\underbrace{\begin{bmatrix} I_x^1 & I_y^1 \\ I_x^2 & I_y^2 \\ \vdots & \\ I_x^n & I_y^n \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{\tilde{u}} = - \underbrace{\begin{bmatrix} I_t^1 \\ I_t^2 \\ \vdots \\ I_t^n \end{bmatrix}}_{b}$$

this is overdetermined, so we solve by least squares:

$$\tilde{u} = -(A^TA)^{-1}A^Tb$$

this cannot be solved exactly if A^TA is singular (rank not 2), e.g.,

at an edge, $A^TA = \begin{bmatrix} \sum {I_x^i}^2 & \sum I_x^i I_y^i \\ \sum I_x^i I_y^i & \sum {I_y^i}^2 \end{bmatrix}$ will only have one non-zero entry, and thus be of rank 1

A^TA is called the __second moment matrix__

No.

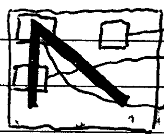
DATE . .

how do we find corresponding points generally, across very different views, e.g., to stitch panoramas together?

<u>Keypoint matching</u> procedure:

1) identify interesting points (called "corners", even when not literally corners)
2) extract feature descriptors for patches surrounding chosen points
3) Match points between the two images

Main idea behind <u>corner detection</u> algorithms: find patches where movement in any direction changes the patch



"flat" region: no change in any direction
"edge": no change when moving parallel to edge
"corner": significant change in all directions

naive approach (loop over patches and shifts, and for each patch, check if "error surface" looks funnel-shaped, i.e., no error for no movement, high error for any movement) is too slow

error surface: $E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$

for window $w(x,y)$ and shift (u,v)

(window may be weighted, e.g., Gaussian centered at $(0,0)$)

optimization: approximate error surface $E(u,v)$ using Taylor expansion centered at $(0,0)$

$$E(u,v) \approx \underbrace{E(0,0)}_{\text{always } 0} + [u\ v]\underbrace{\begin{bmatrix} E_u(0,0) \\ E_v(0,0)\end{bmatrix}}_{\text{always } 0} + \frac{1}{2}[u\ v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0)\end{bmatrix}\begin{bmatrix} u \\ v\end{bmatrix}$$

$$\approx [u\ v]\, M \begin{bmatrix} u \\ v\end{bmatrix} \quad \text{for } M = \sum_{x,y} w(x,y)\begin{bmatrix} I_x(x,y)^2 & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y(x,y)^2\end{bmatrix}$$

M is a second moment matrix, as in optical flow

once $E(u,v)$ is known, we look at how funnel-shaped it is:

$[u\ v]\, M \begin{bmatrix} u \\ v\end{bmatrix} = \text{const}$ is the equation of an ellipse

$\therefore$ we find the axes lengths by eigendecomposition:

$$M = R^{-1}\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2\end{bmatrix} R$$

we want to find places where λ_1, λ_2 are large (indicating the surface is more narrowly funnel-shaped)

we want to avoid places where $\lambda_1 >> \lambda_2$ (indicating ridges)

example response function: $\det(M) - \alpha\, \mathrm{tr}(M)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1+\lambda_2)^2$

for hyperparameter α (larger response is better) after the response $R(x,y)$ has been calculated, we apply thresholding then non-maximum suppression to find individual maxima of $R(x,y)$ — these are the corners!

this entire procedure is called the Harris detector

it is invariant to affine intensity changes ($aI+b$) because only the derivatives are used (scaling factor a may have some effect due to thresholding)

it is invariant to translation and rotation (eigenvalues of M are invariant to rotation)

it is not invariant to scale (solve by computing corners at multiple scales and computing the max)

once corners are known, compute a feature descriptor for each corner

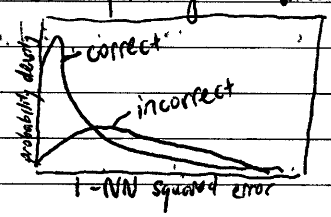
use the gradient at the corner to orient the descriptor (for rotational invariance)

example descriptor: MOPS (multi-scale oriented patches)

take a 40x40 patch, normalize ($I' = (I-\mu)/\sigma$), then downsample to 8x8 (for invariance to small changes)

created in response to SIFT, which was patented, and works almost as well

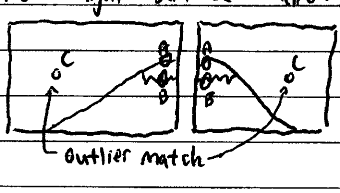
finding matches between images (given descriptors) is hard: just using the 1-NN for each point gives low recall and precision



solution: only choose a match if it is clearly better than all alternatives, i.e. look at the ratio of the 1-NN error to the 2-NN error

this only works if you assume there is at most one correct match

but there might still be outliers, which will ruin homographies



solution: RANSAC (random sample consensus)

1) select a subset of the matches (randomly) and compute a transformation T based on them

2) compute the inliers of T: ($p_i'^T F p_i < \varepsilon$ for 8-point algorithm, $\|p_i' - Hp_i\|_2^2 < \varepsilon$ for homography)

3) repeat, tracking the best inliers, then recompute F or H using those inliers

No.

DATE . .

<u>Camera calibration</u> problem: finding camera parameters — intrinsic (focal length, aspect ratio, etc) and extrinsic (pose) from images and known 3D points (or calibration objects)

simple approach: place a known object in the scene, take a picture, find correspondences, then solve the resulting system for the projection matrix (Π)

$$\underbrace{\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}}_{\text{known}} \sim \underbrace{\begin{bmatrix} M_{00} & \cdots & M_{03} \\ M_{10} & \cdots & M_{13} \\ M_{20} & \cdots & M_{23} \end{bmatrix}}_{\text{projection matrix } (\Pi)} \underbrace{\begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}}_{\text{known}}$$

Π can map any 3D point to its image

Π^{-1} can map a pixel to a ray in the 3D world

but this approach doesn't tell us particular parameters, and it muddles the intrinsic and extrinsic parameters

better approach: solve for specific parameters

decomposition of Π: $\Pi = \begin{bmatrix} \text{intrinsics} \end{bmatrix} \begin{bmatrix} \text{projection} \end{bmatrix} \begin{bmatrix} \text{rotation} \end{bmatrix} \begin{bmatrix} \text{translation} \end{bmatrix}$

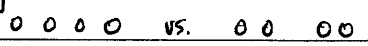
solved via non-linear optimization

but calibration is annoying, so often we assume things about the intrinsics (or get them from the EXIF tag)

focal length is the most important intrinsic (and varies with zoom)

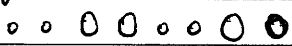
<u>principles of grouping</u> describe how humans naturally perceive objects in organized patterns

proximity:

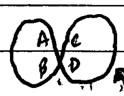


can be in competition:

similarity:

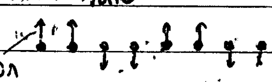


closed form:

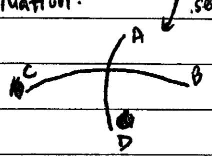


notice these seem to conflict

common fate:



continuation:



common region:

induced grouping:

← spacing is uniform

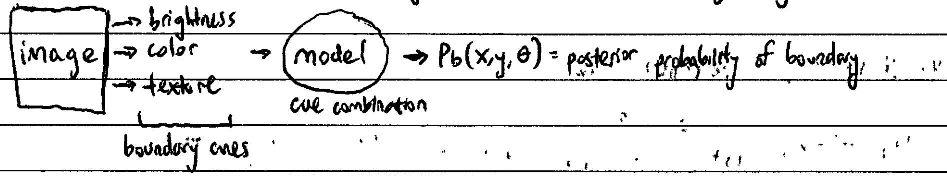
← grouping is caused by this row

No.

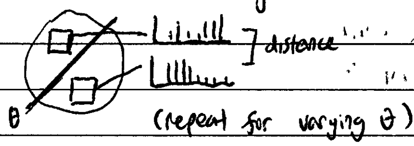
DATE . .

there are also principles of figure-ground organization (distinguishing figure from background)
smaller size, convex shape, symmetry, and high contrast are associated with figures

how can we predict whether a small region is a boundary using only local information?



for brightness and color, we can use filters
for texture, we use histograms of textons then compute χ^2 distance across boundary



we can learn a regression model to combine the cues together
for training, we use a segmentation dataset like BSDS (2001 dataset with human-made ground truths)

alternative approach: model image as graph
each pixel is a vertex, edges connect adjacent vertices, and weights represent similarity
goal: partition graph so in-group similarity is high but between-group similarity is low

minimize this

$$Ncut(A, B) = \frac{cut(A,B)}{vol(A)} + \frac{cut(B,A)}{vol(B)} \qquad cut(A, \bar{A}) = \sum_{i \in A, j \in \bar{A}} S_{ij}$$

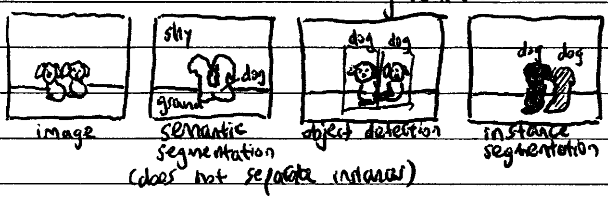
(normalized cut)

sum of degrees: $vol(A) = \sum_{i \in A} d_i$ for $d_i = \sum_j S_{ij}$

similarity/weight matrix (for each pixel, weights are only computed for nearby pixels)

exact discrete solution is NP-hard, but a good approximation can be found

computer vision tasks related to segmentation:



ideas for using CNNs for semantic segmentation:
sliding window that predicts label for each pixel from local context (inefficient)
fully convolutional net that takes in entire image and outputs same-sized label array
can optimize by downsampling and upsampling

No.

DATE . .

transpose convolution: learnable upsampling

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay+bz \\ ax+by+cz \\ bx+cy+dz \\ cx+dy \end{bmatrix}$$

1D conv, size=3, stride=1, padding=1

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay+bx \\ az+by+cx \\ bz+cy+dx \\ cz+dy \\ dz \end{bmatrix}$$

1D conv transpose, stride=1

R-CNN: 2014 object detection system

image $\xrightarrow{\text{proposal method}}$ regions of interest (~2000) → CNN → SVM, Reg; → CNN → SVM, Reg; ⋮ → CNN → SVM, Reg

SVMs for classification

regressions for bounding box fitting

Fast R-CNN: 2015 improvement (trains single large CNN)

Faster R-CNN: learns region proposals directly from CNN features

Mask R-CNN: state-of-the-art instance segmentation method (2017)

basically Faster R-CNN, but with a branch that predicts a segmentation mask