# Computer Vision

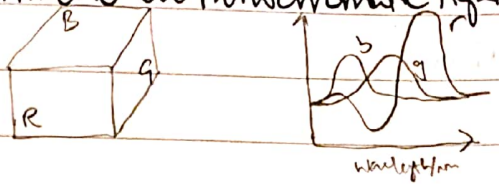
(1) Light sum or difference. How would that change?

Add: Colors add (combine) by adding color spectra (light adds to create black).

Subtract: Colors combine by multiplying color spectra. (pigments remove color from incident white light).

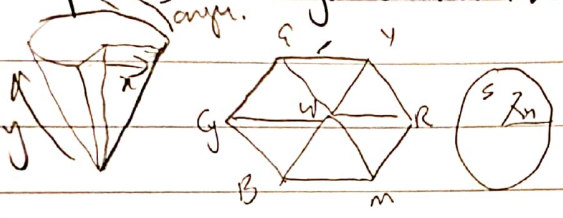
(2) Different between RGB & HSV:

RGB: Primaries are monochromatic) lights, subtractive matching for some wavelength.

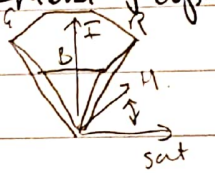



RGB cube on corner.

HSV: Perceptually meaningful dimensions: Hue (angle), saturation (x)




Intensity (y).




(3) Different Binary & Grayscale:

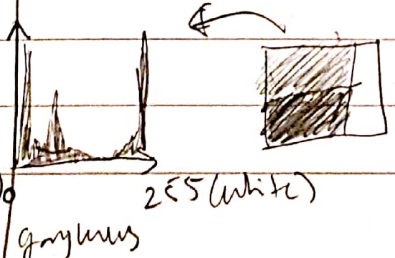
Binary: represents black (1) & white as (0) in a [p x q] matrix.

Grayscale: rep image as 0 → 255 (white) [(0) black]. [p x q] matrix
(intensity)

(4) Creating a histogram (numerical):

Histogram: provides the frequency of the brightness (intensity value) in an image.

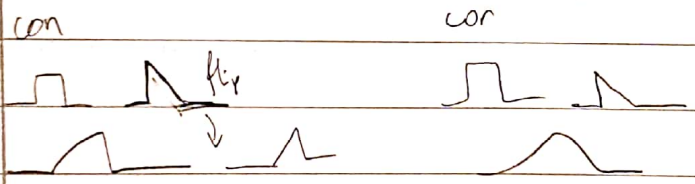
[distribution of gray levels in an image] & [how frequently each gray level occurs]



(5) Define a filter:
Filters form a new image whose pixels values are transformed from the original pixel value.

(6) Apply a filter:
[look at examples & apply formulas]

(7) Difference correlation & convolution: measure similarity between 2 signals
same → correlation: (**) is a measure of similarity between 2 signals.
flip 180° → convolution: (*) is a linear operator on a signal which modifies it. (filtering)
(used for blurring, sharpening, edge detection)




the amount of overlap of one function as shifted over another function.

(8) How do sharpen an image?
original + details = sharpen.

$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + a \left[ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right]$$

(blurred with box filter)

original * $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$ = [ ] shifted by 1 pixel

(a) Derivatives: (forward)(backward)(center).

Back: $\frac{df}{dx} = f(x) - f(x-1) = f'(x) \rightarrow [0 \;\; 1 \;\; -1]$

forward: $\frac{df}{dx} = f(x) - f(x+1) = f'(x) \rightarrow [-1 \;\; 1 \;\; 0]$

centre: $\frac{df}{dx} = f(x+1) - f(x-1) = f'(x) \rightarrow [1 \;\; 0 \;\; -1]$

(b) Enwise derivatives

Gradient vector: $\nabla f(x,y) = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f'_x \\ f'_y \end{bmatrix}$

gradient magnitud: $|\nabla f(x,y)| = \sqrt{f'^2_x + f'^2_y}$

gradient direction: $\theta = \tan^{-1}\left(\frac{f'_y}{f'_x}\right) \cdot \tan^{-1} \Rightarrow \begin{bmatrix} f'_y \\ f'_x \end{bmatrix}$

$\frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ → gradient di

$\frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$ → grading mea

(c) Noise affecting derivatives:

Noise loses the edge, because sommen change occuring ∴

Results in pixels look very different from neighbour.

(Smoothing helps by forcing neighboring pixels to look similar)

mean: $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ gussian: $\frac{1}{16}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}[1\;2\;1]$ (smoothin + derivative)

→ blur image & reduce contrast (smooth surface, low mean)

(d) Best filter for which noise? (mean → box filter)

(i) Salt & pepper noise → MEDIAN FILTER.

(ii) poor resolution → super resolution.

(e) marked up → in-painting

(13) Effect of smoothing (size window)
Reduces noise but also blurs image → size of window inc ∴ blur inc.

(14) Criteria for edge detector:
(1) Good detection: minimize probability of false positive and false negatives
(spurious edges by noise) (missing real edges)
(2) Good localization: edges detected must be as close as possible to true edges
(3) Single response: return one point only for each true edge point.
(minimize no. of local maxima around true edge)

(15) Canny thresholds
Canny uses 2 thresholds: (upper & lower), If a pixel higher than upper threshold then it is an edge pixel: if lower than low threshold then rejected as an edge.

(16) Size of gaussians
σ (gaussian kernel spread/size) → σ large then large scale edges detected, small σ detects fine features.

(17) Hough vs Ransac for line detection.
Hough can detect lines, circles and other structures only if the parametric eqn is known (robust detection under noise & partial occlusion)
Pros (simple, easy implementation, missing & occluded is handled, adaptive to many)
cons (complex computation for objects with many parameters, looks for one single type of object, can be fooled by apparent lines, length & position of line segment cannot be determined, colinear line segments cannot be separated).

RANSAC (random sample consensus):
divides data into inliers & outliers and yields estimate computed from minimal set of inliers. estimate: least line with most inliers & least outliers

Pros: (General method suited for a wide range of model fitting problems, easy to implement & easy to calculate failure rate). $(1-w^n)^k$ (n = points needed for hypothesis (2 for line), w = fraction of inlier pts) (k = samples chosen)

Cons: (only handles a moderate % of outliers without extra bookkeeping, many real problems have high rate of outliers)

(18) Can they be used for things other than lines? why?
Though transform yes! if you give the parametric equations.
Ransac yes, it estimates the parameters of a model by random sampling.

(19) Derivatives & corners
at a corner the image gradient has atleast 2 dominant directions
Corners are repeatable & distinctive
- locality: easily recognize the point by looking through small window.
(Localization) - shifting the windows in any direction should give a large change in intensity
corner: $\sum I_x^2$ large & $\sum I_y^2$ large.

(Find distance) $E(u,v) = \sum_{x,y} w(x,y)\,[I(x+u, y+v) - I(x,y)]^2$

$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \text{ (Symmetric.)}$$

Corner response invariant: $M = g(\sigma)^* \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$

(20) Harris detector (properties) process.

1. Translation invariant
2. Rotation invariant
3. NOT scale invariant.

1. image derivatives
2. square derivatives
3. gaussian filter $g(\sigma_I)$

4. Corner response fct (strong cri
$\theta = \det[M(\sigma_I, \sigma_D)] - \alpha[\text{trace}[M(\sigma_I, \sigma_D)]^2$
$= g(I_x^2)\, g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$

5. Non max suppression.

(21) SIFT

Finds the local maximum using difference of gaussian in space & scale.
(captures keypoints at varying scales) → same interest points independently in each scaled layer.

- most blurred image associate with the keypoints scale [tells us the keypoint region]
- to become rotation invariant, rotate the gradient directions & locations by keypoint orientation.
- create array of orientation histograms to put the rotated gradients into their local orientation histogram. (8 orientation bins = 128 vector)
  [(invariant to rotation because worked with rotated descriptor]
  (invariant to scale worked with scaled image in DOG)
- compare each vector in A to vector in B to find matching keypoints (Euclidean vector for high contrast).
  very large illumination gradients → all values $\leq 0.2$ then normalize.
  (invariant to illumination)

(22) Agglomerative cluster.

Clustering: group together similar data points & represent them with a single token.

- Reduce large amt of data, patches based compression/division,
- represent a large continuous vector with the cluster number.

(Histogram of texture, color, sift vectors)

Segment image into different regions.

predict image in same cluster has same color.

Agglomerative clustering: start each point as its own cluster & iteratively merge the closest cluster.

(1) every point is own cluster.

(2) find most similar pair → merge

(3) Repeat

Any distance, mean distance, min distance.
(single link)
can set threshold → once above threshold stop clustering, min dist between clusters
(Group & splitting)

Compute Link → mean distance points (right clusters).

(23) k means cluster

(1) Initialize cluster centers $(c_1, c_2 \dots c_k)$.

(2) Assign each point to nearest cluster (compute $\delta^t$) → get the assignment for each $x_j$ point to cluster $c_i$ at iteration $t$.

$$\delta^t = \arg\min_{\delta} \frac{1}{N} \sum_j \sum_i^k \delta_{ij}^{t-1} (c_i^{t-1} - x_j)^2 \longleftarrow \text{Euclidean distance centroid}$$

(3)

(4) Compute $c^t$: update cluster centers as the mean of the points.

$$c^t = \arg\min \frac{1}{N} \sum_j \sum_i^k \delta_{ij}^t (c_i^{t-1} - x_j)^2.$$

(24) K-mean vs mean shift

K-mean:
- P: finds cluster centres that minimize conditional variance. (Best for spherical data)
- P: Simple and fast. (Slow)
- C: must choose k — prone to local minima
- C: sensitive to outliers — All clusters have same parameters (distance measured in same radius)

meanshift:
- P: application independent
- P: model free, does not assume any prior shape
- P: single parameter (window size).
- P: finds variable no. of modes, robust to outliers.
- C: output depends on window size
- C: window size is not trivial
- C: computationally complex
- C: does not scale well with dimensions of feature space

(25) Pipeline for object recognition:

Apply a prediction function to a feature representation of the image to get desired output: f(apple) = apple

Training images → image features → training → learned classifier
training labels

testing → image features → learned classifier → prediction

image features: color, shape, texture
(gradient HOG) (PCA) (filter banks)

(nearest neighbour)

(bag of words)

(26) BoW: why weight?

(take a bunch of images extract features and build a dictionary of visual words.
[Given a new image extract features & build histogram for each feature find closest visual word.]

Weights because some visual words are more discriminative than others (the higher the frequency of the document the word appears in the less useful it is).

(27) Camera model parameters:

intrinsic assumptions:
- [unit aspect ratio
- optical center (0,0)
- no skew]

Extrinsic Assumptions:
- [no rotation
- camera at 0,0,0]

from: $P' = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$

$P' = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = K [I\ 0] P$

$P' = K(I\ 0)P \rightarrow = \begin{bmatrix} \alpha & 0 & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$ → camera.

$= \begin{bmatrix} \alpha & 0 & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

Square pixels: → next: $P' = \begin{bmatrix} \alpha & s & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$

skew 0: → intrinsic parameters

$\begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$ $\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}$

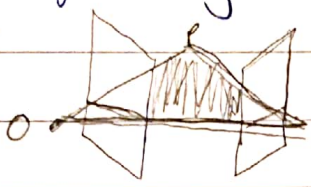
SPUT — 6 DOF (rotation, translation)

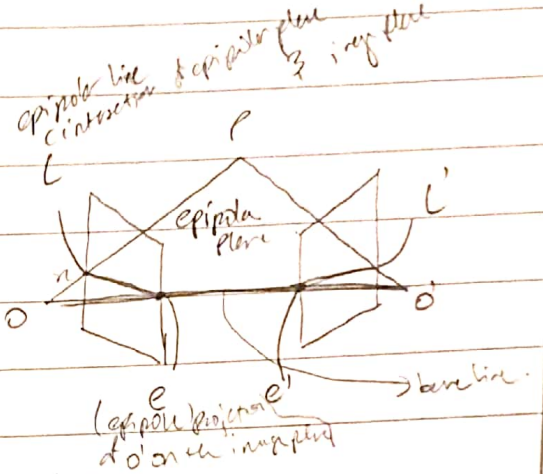
(28) Dof projection matrix

5 6

intrinsic extrinsic.

(29) Epipolar geometry:






Given a point in one image $\xi E x = l'$.

multiply by essential matrix gives epipolar line in 2nd view.

$x'^T E x = 0$ The point $x$ maps to a line $l'$ in right image.

$x'^T l = 0$ The baseline connects the $l$ & $l'$. $e$ & $e'$?

↳ perpendicular An epipole $e$ is a projection of $o'$ on the image plane

$E = R[t_x]$ all epipolar lines in an image intersect at the epipole.

(30) 8-point algh

(1) normalise points

(2) construct $n \times 9$ matrix $A$ from matched pts. $x'^T_m F x_m = 0$

(3) find SVD of $A$.

(4) entries of $F$ are elements of column $V$ corresponding to least singular value.

(5) enforce rank 2 on $F$

(6) unnormalise $F$ ← (SVD)

$\therefore$ pre-processed.

$$\begin{bmatrix} x'_m & y'_m & 1 \end{bmatrix} \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_mx'_m & x_my'_m & x_m & y_mx'_m & y_my'_m & y_m & x'_m & y'_m & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_9 \end{bmatrix} = 0$$

(32) Steps stereo rectification:
(reproject image planes onto a common plane parallel to the line between camera centers)

1) Rotate the right camera by R. [compute E to get R]
2) Rotate (virtually) the left camera so that the epipole is at ∞.
3) Rotate (virtually) right camera so epipole @ ∞.
4) adjust scale.

$T_1 = u_3 \quad T_2 = -u_3 \quad E = [R|T]$

$R_1 = U W V^T \quad R_2 = U W^T V^T$ (choose det = 1)

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

~~rotate both~~: rotate right by R → rotate both images by $R_{rect}$.
Scale Both by H!

(33) Why rectification.
to do stereo matching after...

(34) Assumptions for optical flow:
Recover image motion at each pixel from spatio-temporal image brightness variations (optical flow) → to recover two motion field from two consecutive image frames we want to track optical flow algorithms can help track those.
Assume: the brightness of any object is constant over time
- when projecting the image plane move in a similar manner (velocity smoothness constraint)

(35) derivation?