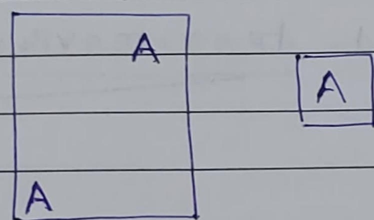
Page :
Date :

12th oct, 2022

lec 1-Am (12th oct, 2022)

* **Interest Point** → sudden variation of intensity values
  ↳ lines, edges, corners, individual points

some points in the image which can form panaromic view (Image stitching).



- template matching
- stereo Image matching
- Image stitching

↳ uses of Interest Point

* Interest Point robust to Affine transformation and photometric variation.
  - photometric ↙ irrespective of illumination variation.
  - Affine ↙ scaling, rotation, translation

→ Interest points have some amount of information (invariant (robust) to affine transformations & Photometric variation).

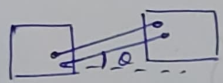
* Why are Interest points useful?
  ↳ stereo matching
  ↳ Panorama ~~stitching~~ stitching

* How to do Panoroma stiching?

Step 1: Extract Features (Generate Interest Points)
Step 2: Match features (not necessary to match all points but majority of points need to match.)

Camlin

rotate before image stitching (if other image rotated by 90°)



→ rotate to bring to horizontal level

→ ignoring points that come outside the majority points

let say majority = 10 to 15

outside this range, points ignored

→ magnitude & phase used to remove redundant points.

Step 3: Align Images.

* Each interest point is described by a feature descriptor.

↳ unique for features

→ Characteristics of good features:

(1) repeatability = same features can be found in several images irrespective of geometric & photometric transformations

(2) saliency = each feature should be unique (very less percentage of error)

(3) compactness & efficiency = important points are found only

↳ feature points less than image pixels

(4) locality = features extracted from a small area (not from large area)

→ features invariant to occlussion (overlapping)

Page :
Date :

(5) efficient = features should be detected very fast

(6) covariant = feature should be detected in their corresponding locations irrespective of geometric / photometric variations.

→ Applications: — Image Alignment
— Motion tracking
— ~~object~~ object recognition

Finding corners:

1st & 2nd derivative = for finding ~~Image~~ Image gradient

~~1st order derivative along X-axis & Y-axis.~~

1st derivative along X-axis, 1st derivative along Y-axis

* SUSAN → can be applied in a noisy image
→ detect edges & corners
→ insensitive to noise
→ No image derivatives
→ circular mask

* circular mask will give isotropic response

give properties invariant to magnitude

circular mask = window

Camlin

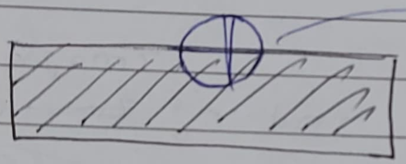
If pixels ~~line~~ lying within the mask have the same brightness as that of the nucleus, then such ~~areas~~ areas are called USAN.

No interest points cannot be generated in USAN region

USAN ~~region~~ region never have lines, edges

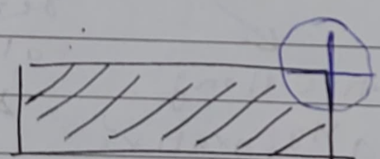
---

* case 2



if some percentage ($\geq 50\%$) of pixels have nucleus value others not in thresholds then this USAN region contains edge.

case 3 *



Almost 25% of pixels have ($<=25\%$) ~~nucleus value~~, then corner point

SUSAN = smallest USAN

↳ smallest area of USAN

→ At the corners, it has the smallest USAN region.

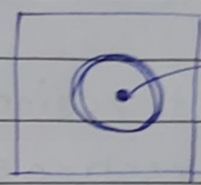
Next 14th Oct, 2023

Implementation

centre pixel $= r_0$

→ USAN region

$$\underbrace{U(r, r_0)}_{\text{total no. of 1 in the images}} = \begin{cases} 1, & I(r) - I(r_0) < t \\ 0, & \text{otherwise} \end{cases}$$



$I(r_0)$ — intensity value of centre point (nucleus)

$T = \frac{3|C|}{4}$ = non-USAN area is $\frac{3}{4}$th of $|C|$, then edge

$T = |C|/2$ = corner

P.T.O ⟹

along direction of edges
~~no in~~ not much
intensity variation

lec2 - AM

* window can be moved in all 8 directions
  ↳ shifting window

Flat regions = Intensity difference = 0 in 8 directions. , no change in all directions

* ~~m~~maximum intensity change = in the corners
* slight intensity change = in the edges

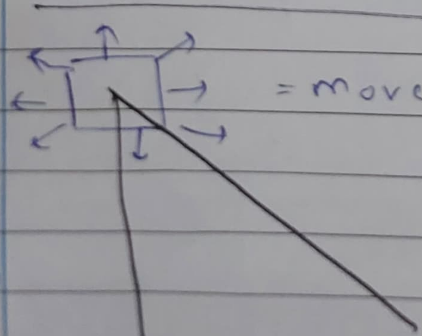
MORAVEC
~~MORAVEC~~ CORNER DETECTOR (window detector)

SSD = sum square difference
↳ calculated for each directions (Total 8 SSD)
↳ find minimum SSD

min. SSD	> threshold	= corner point
	< threshold	= not corner point

Problem: It identify false positive edges, that is some corners are detected as edges.

HARRIS CORNER DETECTION



= move in all directions & check intensity values.

Page :
Date :

u = how much window shifted along x-dirn
v = how much window shifted along y-dirn

equation:

I(x+u, y+v)

u → shift along x-axis

v → shift along y-axis

w(x,y) = window function

I(x,y) = intensity in original location ~~location~~ ~~direction~~

I_x = first derivative along x-axis
I_y = first derivative along y-axis

$\boxed{\text{Corner detection using derivative operation} \Rightarrow \text{Harris corner detection}}$

E(u,v) = change of intensity (at shifted − at current)

lec3-AM (Harris corner detector)

- uses principle of sliding window
- overcomes the problem of Moravec corner detector

* ~~[illegible]~~ constant patch = homogenous region = flat region = 0

corners = E(u,v) is very large.

* large intensity variation in any of directions of corner

largest of $E(u,v)$ = corner points

$E(u,v)$ = Change in intensity

Using first order approx:

$$f(x+u, y+v) = f(x,y) + u f_x(x,y) + v f_y(x,y)$$

$$\boxed{E(u,v) \cong [u, v]\ M \begin{bmatrix} u \\ v \end{bmatrix}}$$

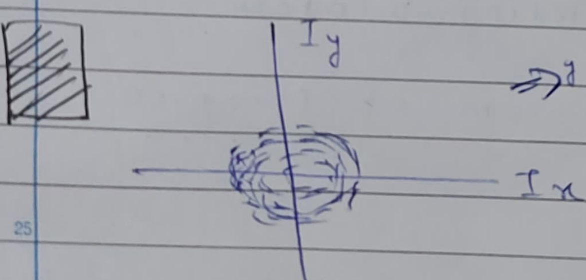
→ Harris detector uses the concept of image gradients (find the gradient along the x-dirn and the gradient along the y-dirn).

Step 1: Find the Gradients

Step 2: Fit the Ellipse

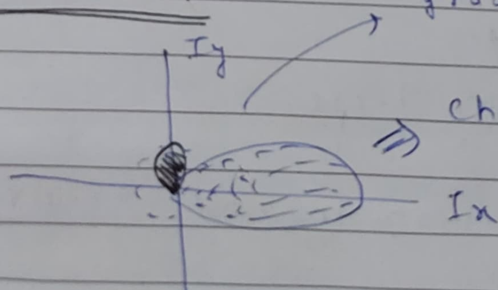
→ Plotting derivatives as 2D points

(1) Flat region



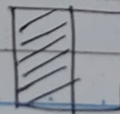
⇒ gradient along the x-dirn & gradient along y-dirn is small and centered around the ~~centre~~ origin

(2) linear edge



gradient along y-dirn is very less.

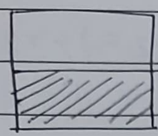
⇒ change in intensity along x-direction

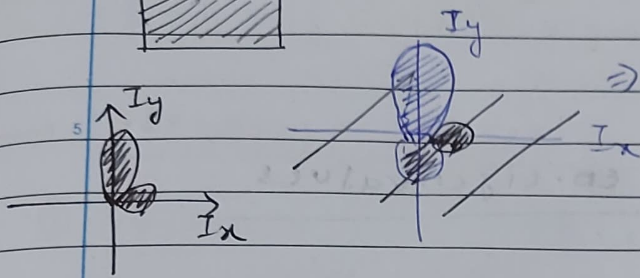


= vertical edge

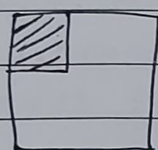
→ Moravec detector intensity diff. between shifted patches

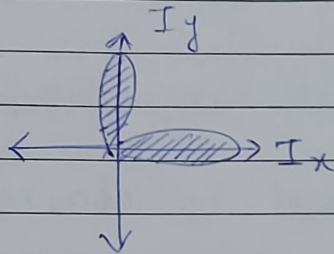
→ Harris corner detector also do same but also play with gradients

(3)  horizontal edge



⇒ Change in intensity along y-dirn

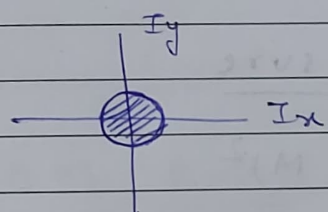
(4) 



⇒ gradients high in both directions (X & Y)

→ Fitting Ellipse to each set of points

(1) Flat:



⇒ we can fit a CIRCLE (NOT ELLIPSE), major & minor axes very small

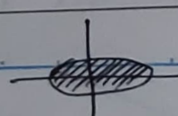
$d_1 \approx d_2$ (both are very small)

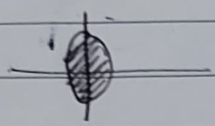
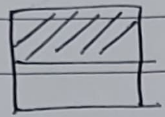
d_1, d_2 = eigen values ~~to the flat region~~ ~~&~~ gradient along X & Y axis.

$$\boxed{\begin{array}{l}\rightarrow \text{major axis will correspond to } d_1 \\ \rightarrow \text{Minor axis will correspond to } d_2\end{array}}$$

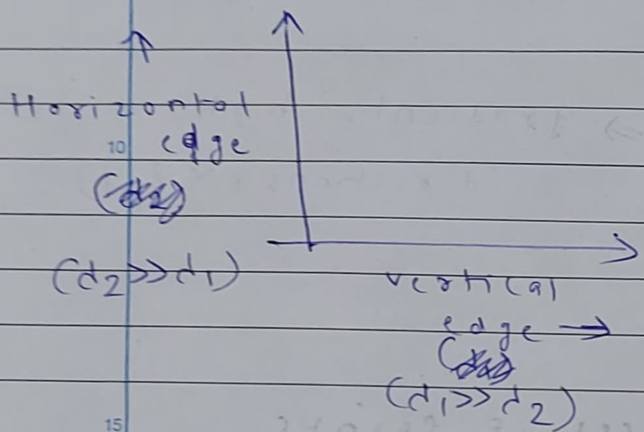
(2) Corner: major & minor axis will be large

$d_1 - d_2$ = large

(3) linear edge:  — d_1 = large, d_2 = small

④   $\Rightarrow d_1 = \text{small},\ d_2 = \text{large}$

$\longrightarrow$ Classification via eigenvalues



$\longrightarrow$ Corner response measure

$$R = \det M - k(\text{trace } M)^2$$

R = corner response
k = constant

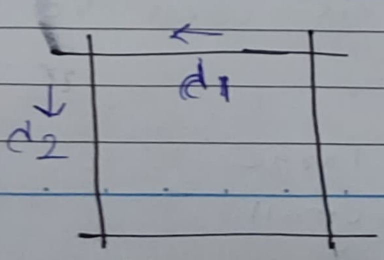
$\det M = d_1 d_2$

$\text{trace } M = d_1 + d_2$

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

$k \approx 0.04 - 0.06$ (depending upon application)

$\rightarrow$ Corner response map



"Higher the response, more is the corner points."

Page :
Date :

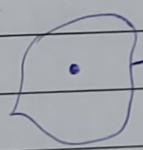
R = small, Flat region
R = high, corner region

→ Advantage of Harris Detector

① irrespective of Affine transformation and illumination variation, able to detect the corners.

② must famous corner detector

→ Taking points of local maxima of R



→ instead of taking this whole region, take the centroid as a point

→ Histogram of gradients

lec 4-AM (HoG)

↳ can determine gradient of image

↳ magnitude and orientation of gradient

↙ HoG most importantly give this

object detection = HoG.

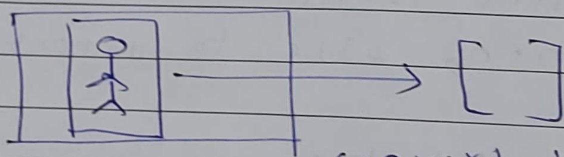
* First find the gradients
* count no. of occurrences of gradient along a particular orientation / direction

↳ Find the direction in which gradients are oriented.

Camlin

→ Steps:

1) normalized train image data set
(set of cropped images of pedestrians in a normal environment)



convert the image into some mathematical domain for recognition purpose

2) [≡] - feature descriptor

should be able to handle various challenges like orientation, illumination condition

3) Binary classifier (ML part)

→ Gradient computation

gradient — magnitude
— orientation

G_x = gradient along X-axis
G_y = gradient along Y-axis

Gradient along Y-dirn, operations along these direction

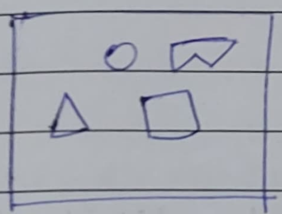
Page :
Date :

Gradient along X-axis along these directions operations done

Ex:

$$\begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline -1 & \alpha & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \qquad \begin{array}{|c|c|c|} \hline & 0 & \\ \hline - & \alpha & - \\ \hline & 0 & \\ \hline \end{array}$$

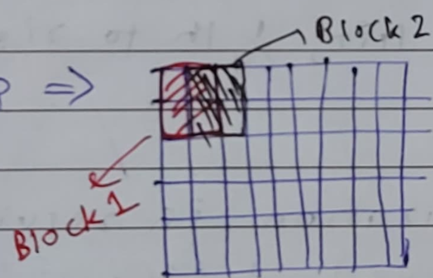
orientation binning



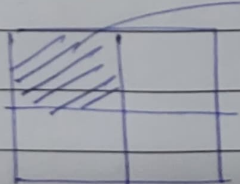
→ crop the image in ratio 1:2

* In ~~HOG~~ HOG, for object recognition, the object should be in ratio 1:2.

* 50% overlap ⇒



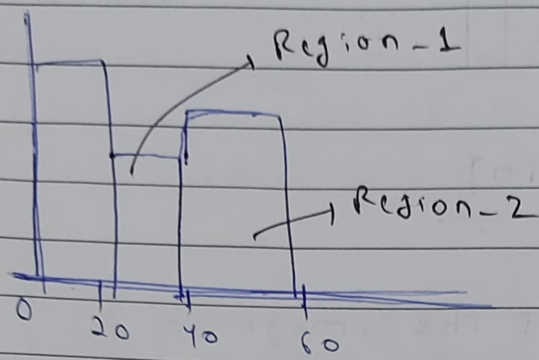
* 1 block should have ~~size~~ cells of 2×2 with ~~pixel 8×8~~. size 8×8.



→ each cell will give gradient orientation

* 0-180° = Divide into 9 bins.

Histogram interpolation



Let say 50 need to be added, divide into 25-25 bins and add to region 1 & 2
45 ⇒ more oriented towards 40 ($\frac{3}{4}$ th towards left, $\frac{1}{4}$ th to right)

important

⟹ Why normalization is required?
Histograms will differ if illumination conditions are different, images taken in dark, light, very bright background.
Normalization is required

Ex: If pixels = 2, 4

divide by $\sqrt{2^2+4^2}$

$$\frac{2}{\sqrt{2^2+4^2}}, \frac{4}{\sqrt{2^2+4^2}}$$

* Before finding gradients, normalize the intensity values.

Camlin

HOG features non-existent in uniform ~~Plates~~ pixel intensity → HOG features non-existent, but when transitioning from black to white (or any colour), HOG exists.

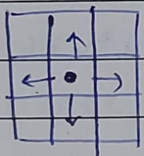
* HOG - YouTube

1) Crop the Image in ratio 1:2

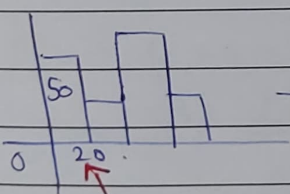
2) resize the cropped image

3) each block of grid = 8×8 pixels

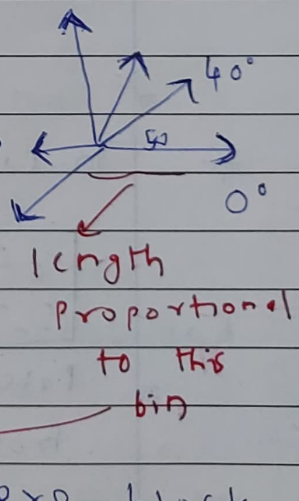
4) For gradient computation, difference in pixels of immediate left, right, up, down



5) Calculate the feature vectors using Gradient magnitude and direction

6) 

→ can be represented in form of ~~arrow~~ arrows →



7) Calculate feature vector for every 8×8 block

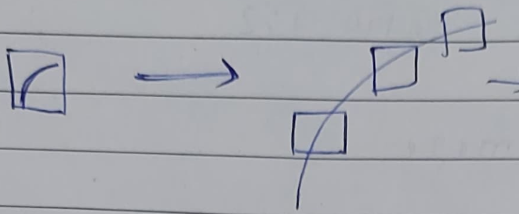
8) Combine feature vectors of 4 adjacent ~~cells~~ blocks 

9) move block to right, do for all blocks in grid

10) After concatenating, normalize

LEC-5-AM (SIFT)

* When the image is scaled, corner detection algorithms do not work that well.

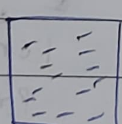


→ if image scaled up, the corner look more like an edge instead of corner

* David Lowe → SIFT
  ↓
  scale invariant feature transform

* Differnt < Background, Rotation, Scale } Despite this, SIFT can get features

* In corner detection / SIFT:



Patches from image → Mathematical transform → Feature vector

{ we compare these FV }

* Canny edge detector = to detect edges
  SURF, GLOH = feature extraction

Page :
Date :

* SIFT Algorithm invariant to :
  → Image Brightness and contrast
  → occlussion (overlapping)

* SIFT $\xrightarrow[\text{used in}]{\text{can be}}$ Panaroma stitching

* SIFT can also be used for SLAM (Simultaneous Localization and mapping) for Indoor navigation of robots.

* Dence SIFT ⇒ variation of SIFT
  ↳ Dence sampling

Lec6-AM (machine learning)

→ ML = learn something from data and then use data, give output

→ ML = give O/P based on the sample data we provide (training data), data provided to ML (testing data)

→ M/c learn on the data provided without any human intervention

→ Works on prediction concepts.

Page :
Date :

⇒ we can provide training data along with Labels to the m/c

ML → unsupervised

ML → semi-supervised

ML → supervised

→ AI = create m/c that mimic human behavior ~~behavior~~

→ ML = no programming needed & human intervention

→ Problems in ML due to
→ Lack of data
→ Lack of access to data
→ Data bias
→ Privacy Problem
→ Badly chosen tasks and Algorithms
→ Poorly recognized features
→ lack of resources
→ evolution problems

→ steps in ML:

Raw data = categorize the data

→ Training set contains larger proportion of data (if 100%, then 70% = training set, 30% = testing set)

Camlin

→ more and more data for training = so as to improve the accuracy and

SVM – 1990's

Support vector m/c
non-probabilistic binary linear classifier

→ Points with same quality should be kept in one class & those with different quality in another class.

→ ~~Boundary~~ Boundary dividing the 2 classes = decision boundary & Plane dividing the 2 classes = hyper plane

→ Applies the statistics of ~~hyper~~ hyper plane

→ Data points in 1 subset = called support vectors

SVM
linear | non-linear

→ Hyper plane must maximize the margin ($d^+ + d^-$)

→ SVM is used in cases ⇒ where no. of features is greater than no. of samples.

→ can be used in 2-classification / multi-classification problem.

Camlin

According to dimension of features, kernel functions are classified

Kernel functions in SVM

→ **linear kernel function**

set of mathematical functions = Kernel

takes data as I/P & transform into required form

→ **Gaussian radial basis function**

for non-linear data

→ **Polynomial kernel function**

Performance → even works well with unstructured & semi-structured data

- SVM not solved for local optima
- overfitting less in SVM
- better results compared to Neural N/W
- longer time for larger dataset

→ SVM less computation, can find only limited set of Patterns

Page :
Date :

→ KNN = complex Patterns

→ SVM
- handwriting recognition
- cancer ~~diagnosis~~ diagnosis
- detecting stegnography

K-NN (k-nearest neighbour)

- non-parametric Classification technique
- developed in 1951
- can be used for classification/regression both

K = 1, object is assigned to that only 1 class

→ Function is approximated locally

Distance measures
- euclidean
- manhattan

Drawbacks of basic majority voting

→ classification occurs when class distribution is skewed.

→ Examples of more dominant class tend to dominate new class.

→ Advantages of K-NN
- Quick for calculation purpose
- simple algorithm to interpret
- Useful Algorithm for regression
- Accuracy is high

Naive bayes = Probabilities used.

Page:
Date:

→ SVM handles outliers better than KNN

→ if Training data larger than no. of features, K-NN better than SVM.

→ Applications of K-NN
- → Text Mining
- → Agriculture
- → finance
- → Facial recognition.