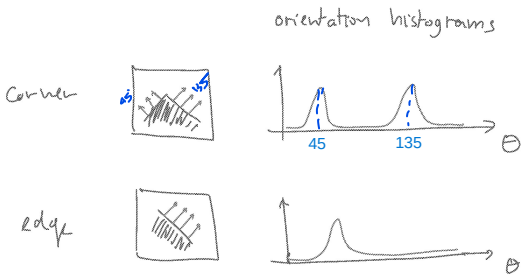
## Corner detection



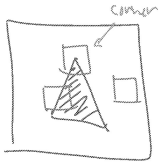

Corner = more than one direction in orientation histogram

To detect a corner, we take a window of appropriate size in the image and then compute the gradients in the window. If there is more than one direction of gradients, it means there is a corner in that window.

We see that in case of edges, there is only one peak in the histogram as there is only one direction of gradients

## Algorithm outline

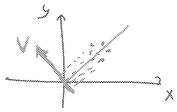
1) Find correlation matrix of gradients in local window
2) Find eigen values of correlation matrix
3) detect corner in window if eigenvalues are sufficiently large.




## Principal component analysis (PCA)

Find direction $v$ s.t. projection of $\{g_i\}$ onto $v$ is minimized:

$$E(v) = \sum_i (g_i \cdot v)^2 = \sum_i (g_i^T v)(g_i^T v)$$




$$= \sum_i (v^T g_i)(g_i^T v) = \sum_i v^T g_i g_i^T v$$

$$= v^T \left( \sum_i g_i g_i^T \right) v = v^T C v$$

($g_i$: $2 \times 1$, $g_i^T$: $1 \times 2$, $C$: $2 \times 2$)

Correlation matrix

additional directions minimize projection s.t. being orthogonal to previous directions.

Note: min instead of max

Suppose we have a bunch of vectors and we wanted to find a vector with direction such that the projection of the vectors onto that vector in that direction would be minimized. So, we have to find v. The direction of the vector v is the principal direction.

Principal direction is one which will minimize/maximize the projection

Note - projection of one vector onto another is done by dot product.

Here, we do square to take the absolute values (because some would be +ve and some would be -ve vector, when we some without sqaure it woufd cancel each othere threby afeecting the output

g.v = g^TV-->because we anyways do row x column-->elementwise multiplication anyways

## Correlation matrix /Structure tensor/second moment matrix

$$C = \sum_i g_i g_i^T = \begin{bmatrix} \sum x_i^2 & \sum x_i y_i \\ \sum x_i y_i & \sum y_i^2 \end{bmatrix} \qquad g_i = [x_i, y_i]$$

($g_i^T$: $1 \times 2$, $g_i$: $2 \times 1$, $C$: $2 \times 2$)

$$C = D^T D$$

$$D = \begin{bmatrix} g_1^T \\ \vdots \\ g_M^T \end{bmatrix}$$

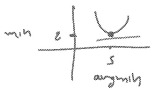
easy method

Do not normalize $g_i$ by subtracting the mean

When we do so, it becomes the Covariance matrix

## Principal component analysis (PCA)

$$\begin{cases} E(v) = v^T C v \\ v^* = \arg\min_v E(v) \end{cases} \qquad C = \sum_i g_i g_i^T$$




We will have to find the arguement value v which would minimize the function E(v)
if the min value of E(V) is 2 , the value of v at the point of E(v) = 2 is 5 --->thats the arg min value..
we look for the size of the projection to be small

$$\Rightarrow \nabla E(v) = 0 \quad \left\{ \frac{\partial}{\partial x} E = 0, \ \frac{\partial}{\partial y} E = 0 \right.$$

$$2Cv = 0 \Rightarrow$$ Solution is eigen vector belonging to smallest eigenvalue

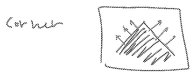
Correlation matrix is built from the gradients of the intensities of the image

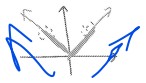
Formula---It can be proved

eigenvalue = variance in corresponding principal direction

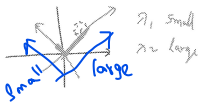
we dont use the eigen vectors, we only need the eigen values as they give the variance in corresponding direction

## Corner detection




Both will be large, because in both directions, there are gradient vectors and hence the projected vector cant be minimized --->No direction of minimal projection


The eigen value will be small in one direction and large in the other



Both are small because the gradients are zero

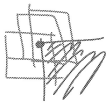
$$\Rightarrow \text{corner if } \lambda_1 \cdot \lambda_2 > \tau$$

## Corner detection summary

1) Scan image T-B, L-R, at each pixel select a local neighborhood.
2) Build correlation matrix
$$C = \sum_i g_i g_i^T$$
3) compute eigenvalues of $C$
4) Detect corner if $\lambda_1 \cdot \lambda_2 > \tau$

## Non-maximum suppression

1) compute $\lambda_1 \cdot \lambda_2$ for all windows
2) Select windows with $\lambda_1 \cdot \lambda_2 > \tau$ and sort in decreasing order
3) Select the top of the list as corner, and delete all other corners in its neighborhood from the list
4) Stop once detecting x% of the points as corners



- overlapping windows are needed to account for corners between windows
- The analysis need to be done at multiple scales

The tau value is used to make the sorting less costly, even if there is no tau value, we would still sort the values in decreasing order and would still take the same top values but sorting all of the valus is unnecessary and incurs computation cost

Review image notes

## Harris corner detection

implemented in OpenCV

1) Compute correlation matrix $C$ for window
2) Compute cornerness measure:
$$G(Q) = \underbrace{\det(Q)}_{\lambda_1 \cdot \lambda_2} - k \underbrace{\text{tr}^2(Q)}_{k(\lambda_1+\lambda_2)^2}$$
(trace: $\text{tr}^2$)
3) detect corners where $G(Q)$ is high

We detect a combo of corners and edges because in real life, corners are not perfect corners

$k \in [0, 0.5]$ is a user parameter, eg: 0.04 – 0.15

It is not hyper parameter, dont have to bother much about chagning it, we can just use the default and fix with it

if $k=0$ $G(c)$ detects corners,
if $k=0.5$ $G(c)$ detects edges

## Harris corner detection notes

$$G(c) = \det(c) - k\,tr^2(c)$$

$$= \lambda_1\lambda_2 - k(\lambda_1+\lambda_2)^2$$

$$= (1-2k)\lambda_1\cdot\lambda_2 - k(\lambda_1^2+\lambda_2^2)$$

$(1-2k)\lambda_1\cdot\lambda_2$: $=0$ if $k=0.5$ — corner detection

$-k(\lambda_1^2+\lambda_2^2)$: $=0$ if $k=0$ — edge detection

Alternative Harris:

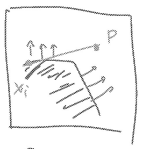
$$G(c) = \det(c) + k\left(\frac{tr(c)}{2}\right)^2$$

## Corner localization

Given that there is a corner in a window find its location.

$$p^* = \arg\min_p E(p)$$

To determine if P is the corner connect each point $x_i$ to p and project the gradient at $x_i$ onto $(x_i - p)$.




The "best" p will minimize the sum of all projections. it will minimize , because the gradients are perpendicular

X is the point on the image, G is the gradient at that point. P is the point whchi we test out as the corner,,,it is a candidate for corner.

The vector thatt connects x an p is determined by X- p. If all the gradients are perpendicular to that vector X-P , then P is the corner point. We project it on X - P because that is where the two vectors intersect and the edges meet for a corner

$\nabla E(p) = 0$

## Corner localization

$$E(p) = \sum_i \left(\nabla I(x_i)\cdot(x_i - p)\right)^2$$

$$= \sum_i (x_i-p)^T \nabla I(x_i)\,\nabla I(x_i)^T (x_i - p)$$

$$= \sum_i (x_i-p)^T \left(\nabla I(x_i)\nabla I(x_i)^T\right)(x_i - p)$$

$1\times 2$ $\quad$ $2\times 1$ $\quad$ $1\times 2$ $\quad$ $2\times 2$ $\quad$ $2\times 1$

E(p) = sum of all projections
project--> dot product of the vectors-->gradient and x-p

## Corner localization

$$\begin{cases} E(p) = \sum_i (x_i - p)^T \left( \nabla I(x_i) \nabla I(x_i)^T \right) (x_i - p) \\ p^* = \underset{p}{\operatorname{argmin}}\, E(p) \end{cases}$$

$$\Rightarrow \nabla E(p) = 0$$

$$-2 \sum_i \left( \nabla I(x_i) \nabla I(x_i)^T \right) (x_i - p) = 0$$

$$\underbrace{\sum \nabla I(x_i) \nabla I(x_i)^T}_{C\ 2\times 2}\ \underbrace{p}_{2\times 1} = \underbrace{\sum \nabla I(x_i) \nabla I(x_i)^T x_i}_{2\times 1}$$

$q$

$$Cp = q$$

$$p = C^{-1} q$$

## Corner localization

$$p^* = \left( \sum_i \nabla I(x_i) \nabla I(x_i)^T \right)^{-1} \sum_i \nabla I(x_i) \nabla I(x_i)^T x_i$$

$$= C^{-1} \sum_i \nabla I(x_i) \nabla I(x_i)^T x_i$$

location of corner

correlation matrix

Since we detected corner in the window

$\lambda_1, \lambda_2 > \tau \Rightarrow C$ must be non-singular

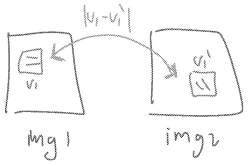
C will be invertble, vbecause we would have chosen correlation matrix only for non -zero eigen values because of the threshold that was chosen....

Eigen value i not zero and so the matrix is not singular -->it should be invertble

## Feature point characterization

* Given a corner we would like to compare it to corners in another image or use it to characterize the image
  ⇒ feature point characterization (find feature vector)

* Applications:
  - matching
  - tracking
  - Correspondence
  - Recognition




matching - matching the same corner in the image
tracking - keep track of the same objecgt in diff images
registration - image registration, trying to align 2 images perfectly by doing some tranformation on the image like rorate and align
recognition - recognize the corner

## Feature point characterization

* Example methods:
  - HoG
  - SIFT
  - SURF
  - Shape context
  - Spin image

* Desired properties:
  - translation invariance → local window
  - rotation invariance → histograms
  - scale invariance → pyramid
  - illumination invariance → gradients

⇒ orientation histograms of gradients at different scales

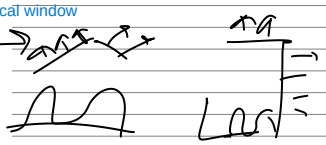
position of tthe feature

checks the neighbor pixels--?local window

size o the image--seeing from far/clloser

dark/bright imge

the property description is a vector of numbers

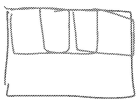


align the histograms to find out if they are same

To capture all 4 this plot helps

## Histogram of oriented gradients (HOG)

1) split each patch into cells (possibly overlapping)
2) create orientation histogram in each cell (using edge or gradient directions, possibly weighted by distance from center or gradient magnitude)
3) Concatenate orientation histograms



e.g. 3×3 cell blocks where each cell has 6×6 pixels

## Scale Invariant Feature Transform (SIFT)

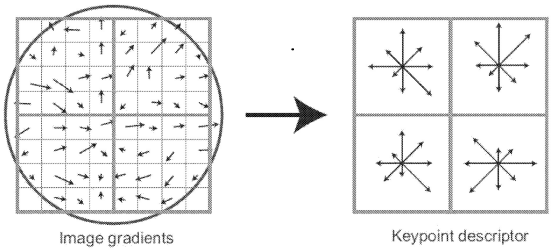


Image gradients

Keypoint descriptor

* Use weighted sum to create orientation histograms in cells, then concatenate.
* Align histogram based on dominant direction (rotation invariance)