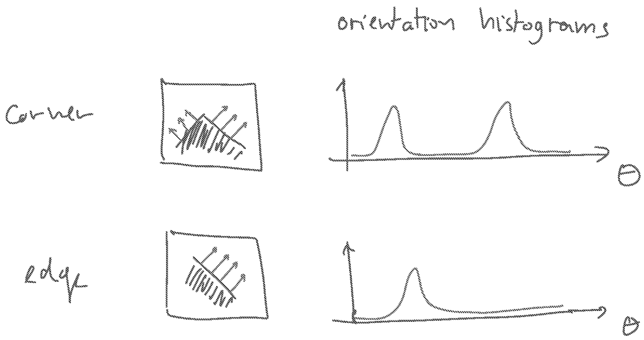
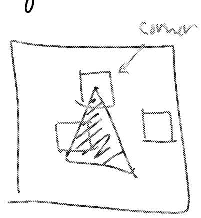
Corner detection



Corner = more than one direction in orientation histogram

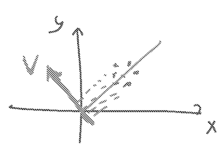
Algorithm outline

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:



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

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

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

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

Note min instead of max

Correlation matrix

$$C = \sum_i \underbrace{\underset{2\times1}{g_i}\,\underset{1\times2}{g_i^T}}_{2\times2} = \begin{bmatrix} \sum x_i^2 & \sum x_i y_i \\ \sum x_i y_i & \sum y_i^2 \end{bmatrix} \qquad g_i = [x_i, y_i]$$

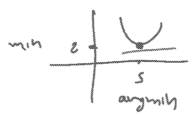
$$C = D^T D$$

$$D = \begin{bmatrix} \text{---}\; g_1^T \;\text{---} \\ \vdots \\ \text{---}\; g_m^T \;\text{---} \end{bmatrix}$$

Do not normalize g_i by subtracting the mean

Principal component analysis (PCA)

$$\begin{cases} E(v) = v^T C\, v \\ v^* = \underset{v}{\operatorname{argmin}}\; E(v) \end{cases} \qquad C = \sum_i g_i g_i^T$$



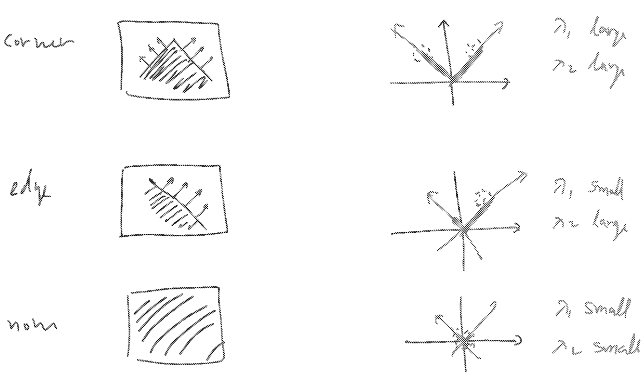
$\Longrightarrow$

$$\nabla E(v) = 0$$

$\cancel{2}\, C v = 0 \Rightarrow$ solution is eigenvector belonging to smallest eigenvalue

eigenvalue = variance in corresponding principal direction

Corner detection



$\Rightarrow$ corner if $\lambda_1 \cdot \lambda_2 > \tau$

Corner detection summary

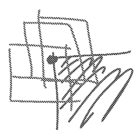
1) Scan image T-B, L-R, at each pixel select a local neighborhood.
2) Build correlation matrix

$$C = \sum_i g_i g_i^T$$

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

Harris corner detection

1) Compute correlation matrix C for window
2) Compute cornerness measure:

$$G(c) = \underbrace{\det(c)}_{\lambda_1 \cdot \lambda_2} - \underbrace{k\, tr^2(C)}_{k(\lambda_1+\lambda_2)^2}$$

3) detect corners where $G(c)$ is high

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

if $k=0$ $G(c)$ detects corners

if $k=0.5$ $G(C)$ detects edges

Harris corner detection notes

$$C(c) = \det(C) - k\,\mathrm{tr}^2(C)$$

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

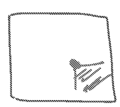
$$= \underbrace{(1-2k)\lambda_1 \cdot \lambda_2}_{\substack{=0 \text{ if } k=0.5 \\ \text{corner detection}}} \underbrace{- k(\lambda_1^2 + \lambda_2^2)}_{\substack{=0 \text{ if } k=0 \\ \text{edge detection}}}$$

Alternative Harris:

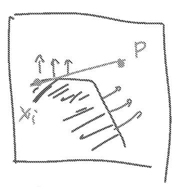
$$C(c) = \det(C) + k\left(\frac{\mathrm{tr}(C)}{2}\right)^2$$

Corner localization

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



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.

Corner localization

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

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

$$= \sum_i \underbrace{(x_i - p)^T}_{1\times 2} \underbrace{\left(\overset{2\times 1}{\nabla I(x_i)} \overset{1\times 2}{\nabla I(x_i)^T}\right)}_{2\times 2} \underbrace{(x_i - p)}_{2\times 1}$$

Corner localization

$$\begin{cases} E(p) = \sum_i (x_i - p)^T \left(\nabla I(x_i)\nabla I(x_i)^T\right)(x_i - p) \\ p^* = \underset{p}{\arg\min}\; E(p) \end{cases}$$

$\Rightarrow$

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

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

$$\underbrace{\sum \nabla I(x_i)\nabla I(x_i)^T}_{C \quad 2\times2} \underbrace{p}_{2\times1} = \underbrace{\sum \nabla I(x_i)\nabla I(x_i)^T x_i}_{2\times1}$$

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

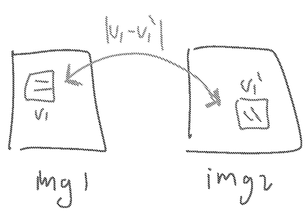
$\lambda_1, \lambda_2 > \tau \Rightarrow C$ must be non-singular

Feature point characterization

* Given a corner we would like to compare it to corners in another image or use it to characterize the image

$\Rightarrow$ feature point characterization (find feature vector)

* Applications:
  - matching
  - tracking
  - Correspondence
  - Recognition



Feature point characterization

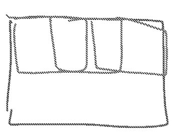
* Example methods:
  - HoG
  - SIFT
  - SURF
  - shape context
  - spin image

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

⇒ orientation histograms at different scales

Histogram of oriented gradients (HOG)

1) split each patch into cells (possibly over-lapping)
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)

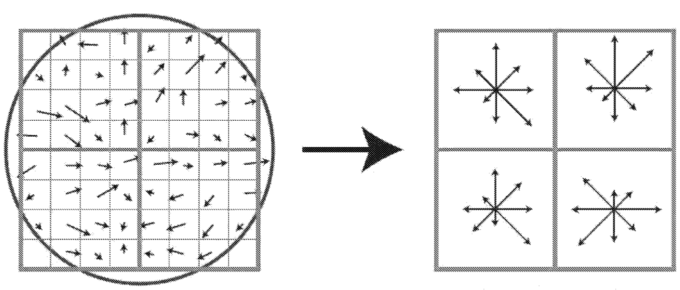
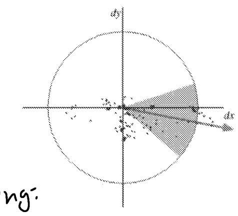


Image gradients   Keypoint descriptor

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

Speeded-up robust features (SURF)

1) obtain oriented responses using Haar wavelet filters
2) Combine wavelet filter outputs using weighted sum
3) Integrate in a sliding cone
4) Find dominant direction



5) Divide each patch to sub-regions and represent each sub-region using:
   $[\Sigma d_x, \Sigma d_y, \Sigma |d_x|, \Sigma |d_y|]$
6) concatenate sub-region representations