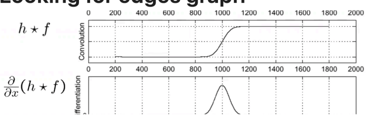
Looking for edges graph

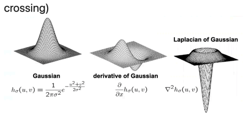


Gaussian derivative (speed things up):

- don't have to differentiate whole thing -diff gausssian * f

- x high -> low
- y low -> high

Take laplacian of gaussian (2nd derivative - zero crossing)



- more sigma smooths it out over wider -> less edges in more detailed areas

1. smoothing
2. edge enhancement

1&2 can be done through derivative of gaussian

Canny edge detection

3. edge localization (threshold) -> binary image
   1. pixel less than t set to 0
   2. others to 1

- non max suppression edge width-> single pixel
  - specific location by gradient direction
- low and high threshold (high starts low finishes edge curves)

Derivatives

- smoothing *negative signs* used to get high response in regions of high contrast
- sum to 0 -> no response in constant regions
- high abs at points of high contrast

edge strength = gradient magnitude

- choose min seam

$\mathbf{M}(i,j) = Energy(i,j) + \min(\mathbf{M}(i-1,j-1), \mathbf{M}(i-1,j), \mathbf{M}(i-1,j+1))$

Harris Corner Detection

Need:

- repeatability: despite geometric transformations - correspondence
- Saliency: small distinctive subset
- compactness/opt: fewer features than pixels
- locality: small area of image (no cluter)

How:

- eigenvalues give us basis vector
  - eigenvalue >> other -> edge
- rotation invariant: by eigen to find new basis rotation
- scale invariant: no -> yes
  - choosing window size: expand radius -> get most activation for each pixel

1. cornerness: M matrix
2. threshhold: keep high
3. local filter for most cornerlike
   1. non-max suppression

Formulas:
R = det(M) - alphatrace(m)^2
R small: flat, R > 0: corner,

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

Blob detection

- edge = ripple, (2nd of guass) -> blob is is where ripples dip lower
  - mag of laplacian will provide scale of blob - smaller laplacian dip wants smaller blob -> (characteristic scale)

trying to do multiple scales

- simple descriptors: raw pixel vectorized (highly sensitive to noise/shifintg) -> SIFT
- Histograms to bin pixels sub-patches accoridng to thier graident orientation (don't vectorize)
  - mag: old determines weight of histo
  - angle: tan
- compute histograms for different patches

Oreint patch by max weight and make this the 0

- less bins -> less orientations = less info
- more bins -> too much info

partially invariant to

- illumination changes
- camera viewpoint
- clutter

$$M = X \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} X^T$$

- find patches that have most similar (lowest SSD)
  - **(Xi – Xj)2 + (Yi – Yj)2** for every distinct pair **(i, j)**
- robustness: distance to best match/distance to second best match
  - =1 is ambiguous
  - lowest: first match looks good
- hypothesize tranformation -> apply trans and see if more match

Clustering algorithms

i. want to find centers

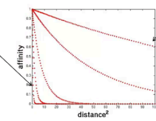
- minimize SSD between all points in cluster

ii. methods

- k means: randomly initilize k clusters -> for each point find closet c -> given points avg for c -> if c changed iterate again (without random)
  - gives local min (converges reasonably in time), sensitive to start/#initial clusters/must cluster all points(outliers suck), detects spherical clusters
- Feature space
  - intensity+position or color
- normalzied cuts: have similar appearance forming parts of an object
  - build graph node for every pixel -> every edge has weight which is similarity (affinity)

$$aff(x,y) = \exp\left\{-\left(\frac{1}{2\sigma_d^2}\right)\|x-y\|^2\right\}$$

Small sigma: group only nearby points



Large sigma: group distant points

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

- exp((-1/(2sigma^2) * ||xi - xj||^2)
- delete links that cross btw segments
  - cut low affinities
  - min cut (summing up all the weights taht you cut) - removal makes graph disconnected
- generalized eigenvalue problem
- pro: does not require model fo data distribution, flexible choice of affinities
- cons: comp high, dense, preference fore balanced partitions (equal weights - or else could favor really small clutsters ex: small object on big background)

Sampling

- sample the 2d space on grid -> quantize each sample (int)
  - one value per pixel
- sample across R, G, B (can be avged together for one image)

Filter (denoise, resize, extract texture, edges, detect patterns)

- enhance image - denoise
  - raw pixel of same image won't be same
- Salt and pepper: white pixels - guassian noise sample from guassian N(u, sigma) more var = more noise

Reducing noise: average of neighbors - expect neighbors to be similar [1, 1, 1, 1, 1]/5 (uniform) [1, 4, 5, 4, 1]/16

H = 1/9 [matrix of 1's]

Correlation filtering: G[i,j] = 1/(2k+1)^2 $\sum_{u=-k}^{k}(\sum_{v=-k}^{k} F[i+u+, j+v])$ G = HxF H is mask produces flipped -> convolution then apply cross (symmetric will output same)

- full = any part of g touches f, same = same size as f, valid = doesn't fall of edge
- boundary (clip filter black, wrap around, copy edge, reflect across edge)
- window size doesn't imapct gaussian -but variance does directly (filter size ~ 6σ) choose window size from this
- remove hf; low pass filter

Runtime: O(n^2m^2) -> 1d G * 1d G = 2d G

O(N) (by seperability outer product)

prop: convolution is linear, can shift

Non-linear filters: median (comp heavy)

low freq: smoothing

high freq: og + (og - smooth = details)

$$Filter_{ij} \propto \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

$$Filter_{ij} \propto \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right) \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{y^2}{2\sigma^2}\right)$$

Prewitt

$$M_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \qquad M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Roberts

$$M_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad M_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$