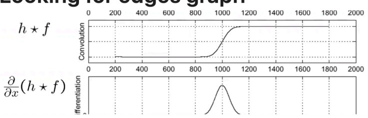
Looking for edges graph

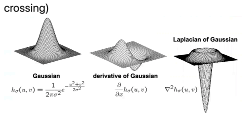


Gaussian derivative (speed things up):

- 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

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

$$p \equiv \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad [R_{3x3} \quad t_{3x1}] \quad X_{4x1}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv M_{3x4} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Pairs of [X,Y,Z] and [u,v] → eqns to constrain M
How do I get [X,Y,Z], [u,v]?

$p_i \times MX_i = 0$

$$M = \begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \end{bmatrix}$$

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \times \begin{bmatrix} m_1^T X_i \\ m_2^T X_i \\ m_3^T X_i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

error = (a*u + b*v + c) / np.sqrt(np.square(a) + np.square(b))

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)

<u>Stereo & Camera Calibration</u>

Given 2 cams + correspondence → find coord for real world point
Estimate intrinsic matrix K
[u v 1] = [K t][x y z 1]
Use u, v ↔ x, y, z correspondences to calculate

Linear Method estimate K:

$p_i = \lambda M X_i,\ \lambda \neq 0 \rightarrow p_i \times MX_i = 0$

Cross product → skew symmetric

$$\begin{bmatrix} 0^T & -X_i^T & v_i X_i^T \\ X_i^T & 0^T & -u_i X_i^T \\ -v_i X_i^T & u_i X_i^T & 0^T \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

11 degs. freedom, ambig. to scale
argmin $||An||_2^2$ s.t. $||n||_2^2 = 1 \rightarrow$

e_vec of $A^T A$ w/ small e_val

In practice:
<u>**1.**</u> Get $K[R\ t]$ from M →
QR decomp of leftmost 3x3 in M → upper triangular matrix * rotational
<u>**2.**</u> Extra correspondences = over constrained
So → set up linear equations
Non-linear opt (ie least squares):

$$\sum ||\text{proj}(MX_i) - [u_i, v_i]^T||_2^2$$

Triangulation: solve for X given M:
Geom: midpoint of viewing rays

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

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i-u, j-v]$$

$$G = H \star F$$

Cross-correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i+u, j+v]$$

$$G = H \otimes F$$

- can't convert directly to pixel lcoation
    - lies btw two pixels -> distribute color amongst neighbor (splatting)
        - a lot of blur
- inverse warping $(x, y) = T^{-1}(x', y')$
    - btw: bilinear in 2d a,b distance is weight to the four points

- $$\min ||Lh||^2 \quad \text{s.t. } ||h||^2 = 1$$
    - h are unknowns (h1 is row)
- equations: $x' = \frac{h_1^t p}{h_3^t p}$
- minimize: $x' -> h_1^t p - x'(h_3^t p) = 0$
- L = $$L = \begin{bmatrix} \ldots\ldots\ldots \\ p_i^T & 0 & -x'_i p_i^T \\ 0 & p_i^T & -y'_i p_i^T \\ \ldots\ldots\ldots \end{bmatrix}$$
- $min(h^t Ah)$ s.t. $||h||^2 = 1$
    - smallest eigenvalue solves

- 4x4 matrix of histograms of pixel gradient angles
Rotate histogram to dominant orientation
Invariant to:rotation, scale ~camera&env. changes
Find matches: find two SIFT with lowest sum of squares distance (SSD), take closest (1 - k)
Robustness rt.: dist to best / dist to 2nd best
→ use matches to hypothesize transformation T

Clustering algorithms

i. want to find centers

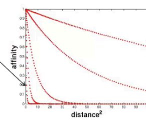
- minimize SSD between all points in cluster

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)

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right) \qquad ||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Reducing noise = average across img.
<u>Correlation Filtering:</u> (G=new, F=old, H=filter)

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \underbrace{H[u,v]}_{\text{Non-uniform weights}} F[i+u, j+v]$$

Smoothes by evening out spike in matrix. (Flips filter bot to top, right to left)

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

0	0	0
0	2	0
0	0	0

\- $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

0	0	0
0	0	1
0	0	0

Blur with Box **Sh**arpening Filter **Sh**ift Left by 1 pi

0	0	0
0	2	0
0	0	0

<- **Sh**arpens the image

Properties of Convolution ---->

Any shift-invariant, linear operation is a convolution (⋆)
Commutative: f ⋆ g = g ⋆ f
Associative: (f ⋆ g) ⋆ h = f ⋆ (g ⋆ h)
Distributes over +: f ⋆ (g + h) = f ⋆ g + f ⋆ h
Scalars factor out: kf ⋆ g = f ⋆ kg = k (f ⋆ g)
Identity (a single one with all zeros):

<u>Grassman's Law</u>
1. If two test lights can be matches with the same set of weights, then they match each other
2. If we scale the test light, then the matches get scaled by the same amount.
3. If we mix two test light, then mixing the matches will match the result (superposition)

<u>Linear Color Space</u>
RGB = Single Wavelength primaries, good for devices but not for perception
CIEXYZ = Y value approximates brightness, project to display: (x,y) = (x/x+y+z),(y/x+y+z)
<u>Non-Linear Color Space</u>
HSV = **H**ue, **S**aturation (Purity, intensity), **V**alue– Nonlinear-reflects topology of colors by coding hue as an angle. More User friendly then the other two.
Distance in color space is a problem that no one is really trying to solve except for **LAB**

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]

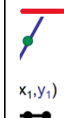
Correlation filtering: G[i,j] = 1/(2k+1)^2 $\sum_{u=-k}^{k}(\sum_{v=-k}^{k} F[i+u, j+v])$ G = HxF H is mask produces flipped -> convolution then apply cross (symmetric will output same)

- 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)

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

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



Start with two points (x_i,y_i)

$$y = Av$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix}$$

$$||y - Av||^2 = \left\|\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} - \begin{bmatrix} mx_1 + b \\ mx_2 + b \end{bmatrix}\right\|^2$$

$$= (y_1 - (mx_1 + b))^2 + (y_2 - (mx_2 + b))^2$$

Roberts

$M_x =$

1	0
0	-1

$M_y =$

0	1
-1	0

Prewitt

$M_x =$

1	0	-1
1	0	-1
1	0	-1

$M_y =$

1	1	1
0	0	0
-1	-1	-1

Sobel

$M_x =$

-1	0	1
-2	0	2
-1	0	1

$M_y =$

-1	-2	-1
0	0	0
1	2	1

$\frac{\partial f(x,y)}{\partial x}$

-1	1

Partial Deriv

1
-1

$\frac{\partial f(x,y)}{\partial y}$

Larger σ = larger scale edges detected
Smaller σ = fine features detected