2d Transform

- Method: finding correspondences
- transformations: translation, rotation, aspect, affine (change shape), perspective
  - p' = T(p) = Mp
  - when p is in R^2 -> 2x2 M
- uniform scaling: same scalar for all components
- non uniform -> new aspect ratio
- sheer

$$x' = x + sh_x * y$$
$$y' = sh_y * x + y$$

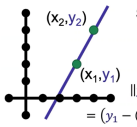
- mirror about y

$$x' = -x$$
$$y' = y$$

- parellel lines remain parallel
- feature based alignment
- eigenvectors are orthonormal
  - symmetric A eigenvector w largest λ max
  - $\frac{x^T Ax}{x^T x}$ unit vectors max num
- non symmetric -> sVD: U∑VT^
  - u rotation eigenvectors of AA^t
  - σ sqrt of A^tA λ's

$$\begin{bmatrix} w' & y' & x' \\ 0 & d & a \\ 0 & e & b \\ 1 & f & c \\ w & y & x \end{bmatrix}$$

- solving least squares invert A find v (m,b) point -> minimizing error
  - by argmin:



Start with two points (x_i, y_i)

$$y = Av$$

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

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

- satisfying:

Solution satisfies $(A^T A)v^* = A^T y$

<u>or</u>

$$v^* = (A^T A)^{-1} A^T y$$

- when A is square, rank < rows or rank < cols
- homo ls -> orthogonal to vectors argmin(||Av||^2) smallest ev from AtA, smallest singular v of A
- for 2 variables need at least 2 eq

ransac - reduce noise

- not true matches -> outlier (minimising lq creates error)
  - look for inliers

For N times

1. select random seed group s points
   1. more points = more robust
2. compute transformation for seed group
3. find inliers to this transformation
   1. point whose d is < t
4. if large then recomputer estimate M on all of inliers
   1. d > inliers accept and refit using all inliers

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotate

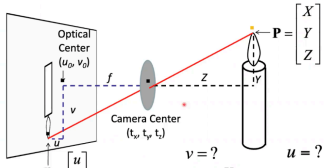
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Shear

- image formation
  - every point on a tree blends its colors all across the film
  - -> barrier known as aperture (only one makes it through) flipped
  - f = focal length, c = center of camera
    - f distance from pinhole and img
  - lines all to vanishing point
- facts about projection
  - line in 3d -> line in 3d
  - parallel 3d -> loses, but converge to centr point
  - distant objects are smaller
- increase aperture (when too dark)

photometric: type, direction, intensity of light, surfaces reflectance property

optical: focal length, fov, aperture, shutter speed



$$\mathbf{P} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

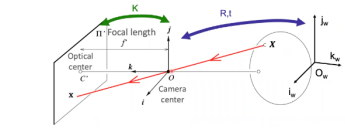
$$\mathbf{p} = \begin{bmatrix} u \\ v \end{bmatrix}$$

$$v = ? \qquad u = ?$$

$$v = f\frac{Y}{Z} + v_0 \qquad u = f\frac{X}{Z} + u_0$$

evi Parikh from Derek Hoiem

$$K[Rt]X$$



amera Properties

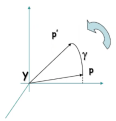
World Properties

x: Image Coordinates: (u,v,1)
K: Intrinsic Matrix (3x3)
R: Rotation (3x3)

- extrinsic ass: no rotation, camera at 000 4th col
- intrinsic ass: opticalcenter000,unit aspe,!skew
- rotations:

Rotation around the coordinate axes, counter-clockwise:

$$\boldsymbol{R} = R_x(\alpha)R_y(\beta)R_z(\gamma)$$



$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5 6

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- QR - decomposition of 3x3 matrix -> finite options of upper tri matrix * rotation
  - to find $K_{3x3}[R_{3x3}, t_{3x1}]$
- overconstrained MX-> not perfect equality
  - parameters with linear model

$$\begin{array}{l} \text{Correspondence 1} \\ \\ \text{Correspondence n} \end{array} \begin{bmatrix} \mathbf{0}^T & X_1^T & -v_1 X_1^T \\ X_1^T & \mathbf{0}^T & -u_1 X_1^T \\ \cdots & \cdots & \cdots \\ \mathbf{0}^T & X_n^T & -v_1 X_n^T \\ X_n^T & \mathbf{0}^T & -u_n X_n^T \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

How do we solve problems of the form

$$\arg\min \|An\|_2^2$$
$$s.t.\ \|n\|_2^2 = 1\ ?$$

Eigenvector of A^TA with smallest eigenvalue

Stereo: given 2 cameras/correspondence

- find: world coordinate for known point

Epipolar Geometry

baseline connects both origins

- epioles where baseline intersects w img planes
- epiploar lines btw point and e -> p' (must be from same light ray)
  - rays given by ep line
- only translation -> ep lines horizontal, e ∞
- forward motion -> ep lines out from optical cent

Calibrated case

- know intrinsic & extrinsic -> set coord to cam 1
  - $M_1 = K[I, 0], M_2 = K'[R, t]$, M produce p
- $K^{-1}p$ producing normailized coordinate p^ with K as identity matrix (canonical view)
- coplanar: Rp^, t, p^ -> $\hat{p}'^T(t \times R\hat{p}) = 0$

$$\hat{p}'^T \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} R\hat{p} = 0$$

- $x^T E y = 0$ E is t_x: essential matrix
  - bc of p^ being normalized
  - $\hat{p}'^T E \hat{p} = 0$ E = $[t_x]R$
  - Ep^ gives eq for ep line for o'
  - E^Tp^ gives ep in o
  - epipoles in nullspace of E: $E^T\hat{e}' = 0$ and $E\hat{e} = 0$

Set: $F = K'^{-T}EK^{-1}$ Then: $p'^T F p = 0$ Fp F^tp' are ep lines w p, p'

Each point gives an equation:

$$[uu', uv', u, vu', vv', v, u', v', 1] \cdot [f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33}] = 0$$

Stack equations to yield **U**:

$$\boldsymbol{U} = \begin{bmatrix} \vdots & & & & & & & & \vdots \\ u_i u_i' & u_i v_i' & u_i & v_i u_i' & v_i v_i' & v_i & u_i' & v_i' & 1 \\ \vdots & & & & & & & & \vdots \end{bmatrix}$$

How to solve for **f** (F unrolled)?:

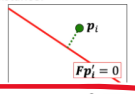
$$\arg\min_{\|f\|=1} \|\boldsymbol{U}\boldsymbol{f}\|_2^2 \rightarrow$$ Eigenvector of $\boldsymbol{U}^T\boldsymbol{U}$ with smallest eigenvalue

Minimizing via U^TU minimizes sum of squared ***algebraic*** distances between points $\boldsymbol{p_i}$ and epipolar lines $\boldsymbol{Fp'}_i$ (or points $\boldsymbol{p'}_i$ and epipolar lines $\boldsymbol{F^T p_i}$):

$$\sum_i (p_i^T F p_i')^2$$

May want to minimize ***geometric*** distance:

$$\sum_i d(p_i, Fp'_i)^2 + d(p'_i, F^T p_i)^2$$



- old*
  - linear independence
  - span: all linear combiantions of set of vectors
  - Ax output is constrained to span of cols of A
  - full rank: one-to-one mapping -> invertible
- Symmetric $A = X^T X$
- rotation matrix: all row/col have l2 norm, lin ind, transpose is inverse, det = 1, eigenvlaue = 1
- eigensystems: lagest eigenvector = vector with largest value

Approximating Laplacian

$$L = \sigma^2 (G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma))$$

(Laplacian)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)

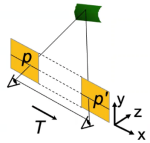
$$p_1 \equiv M_1 X \qquad p_1 \times M_1 X = 0$$
$$p_2 \equiv M_2 X \rightarrow p_2 \times M_2 X = 0 \rightarrow [p_1]_\times M_1 X = 0$$
$$p_2 \times M_2 X = 0 \qquad [p_2]_\times M_2 X = 0$$

- X is only unknown
- calibration -> solve M, triangulation -> solve X

- F is estimate known as weak calibration
- E = $E = K'^T F K$
- from E calc relative rotation/translation
- scene point z direction
- disparity: x - x', inveresly proportional to depth

For each pixel

1. find coreresponding ep line in right
2. search along and find best match (SSD)
3. tri matches to get depth



$$p'^T E p = 0 \qquad E = [t_x]R$$

What's R? $R = I$ **What's t?** $t = [T, 0, 0]$

$$E = [t_x]R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix}$$

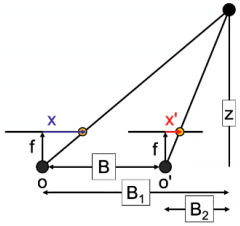
$$[u'\ v'\ 1] \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0 \rightarrow [u'\ v'\ 1] \begin{bmatrix} 0 \\ -T \\ Tv \end{bmatrix} = 0 \rightarrow \begin{array}{l} -Tv' + Tv = 0 \\ Tv = Tv' \end{array}$$

y points are the same

Stereo image rectification

- create virtual planes that only differ by translation
- -> pixel motion is horiz -> 2 homo for each img
- distance min, similarity max
  - sum(l - r)^2 vs

Triangulation



$$\frac{x}{f} = \frac{B_1}{z} \qquad \frac{x'}{f} = \frac{B_2}{z}$$

Subtract them

$$\frac{x - x'}{f} = \frac{B_1 - B_2}{z}$$

$$\underbrace{x - x'}_{\text{Disparity}} = \frac{fB}{z}$$

$$z = Bf/(z-x')$$

- associate pateches with object of interest
- frames of a single shot, faces, background vs what is moving

Gestalt: symmetry, similarity, common fate (coherent motion), proximity, subjective contors (why shape exists)

- whole is other than sum of its parts

segmentation: separate into coherent objects

- top down: pixels same object -> group
- bot up: pixels look similar -> group
- superpixels: similar looking pixels

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

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

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
- normalized cuts: have similar appearance forming parts of an object
- build graph node for every pixel -> every edge has weight which is similarity (affinity)

Line fitting

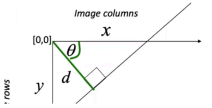
- why fit lines: many objects characterized by presence of straight lines
- problem: extra edge points clutter it, missing lines, noise in edge points

Voting

- all local features vote for all models compatible with it (look at parameters with lots of votes) - noise will make votes but will be as outlier
  - for every edge point
- look for lines that get many votes

Process:

- y=mx + b -> point based on parameters m and b
- point in x,y -> b = -xm + y (line in hough space)
- intersection of lines in hough space = line that passes through two coordinates in image space -> quantize and bin each grid spot for votes
- polar representation:



d: perpendicular distance from line to origin

θ: angle the perpendicular makes with the x-axis

$$x\cos(\theta) + y\sin(\theta) = d$$

- point in img -> sinusoid segment in hough

Algorithm

1. $H[d, (\theta)] = 0$
2. for each edge point I[x,y]
   1. for theta = $[\theta_{min} to \theta_{max}]$
   2. theta = graident at x,y
   3. $x\cos(\theta) + y\sin(\theta) = d$
   4. $H[d, (\theta)] += 1$
3. find (d, theta) where H[] is max
4. detect line given by d

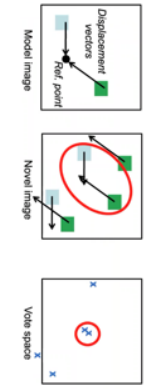
- same d and theta can't vote for multiple lines
- still see peaks in noise voting space -> gradient & orientation
- or give more votes to stronger edges
- or smapling of d, theta to give more or less res

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- now cirlces instead of lines intersecting -> intersecting circles centered at points of hough space -> on edges of circle in img
- unknown radius = cone in hough space
  - can use gradient angle of img space -> hough space line ray
- for circle:

```
For every edge pixel (x,y) :
  For each possible radius value r:
    For each possible gradient direction θ:
      // or use estimated gradient at (x,y)
        a = x – r cos(θ) // column
        b = y – r sin(θ)  // row
        H[a,b,r] += 1
    end
  end
```

- use direciton of edge to reduce para by 1
- minimize irrleavant tokens first
- choose a good grid/ discretization
  - too fine<--->too course
- voting for nearby (smoothing in acc array)

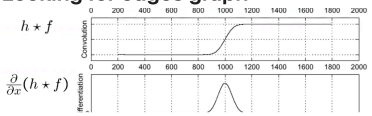


- offline procedure: at each boundary point find r = a - p_{i} a is potential center, r vote
  - dispaclement vectors in table indexed by gradient orientation
  - look for overlaps in r
- not rotation invariant

Pros: all points are processed ind, don't need all points on line, noise points unlikely to contribute big to bin, detect multiple instances of a model in single pass

Cons: more complex shapes for each parameter, non-target shapes offset, quantizations of grid need to bet specified

Looking for edges graph

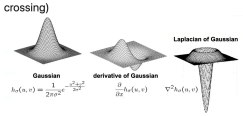


Gaussian derivative (speed things up):

- 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

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

$$\boldsymbol{p} \equiv \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad [\boldsymbol{R}_{3x3} \quad \boldsymbol{t}_{3x1}] \quad \boldsymbol{X}_{4x1}$$

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

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

$$\boldsymbol{p}_i \times \boldsymbol{M}\boldsymbol{X}_i = \boldsymbol{0}$$

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{m}_1^T \\ \boldsymbol{m}_2^T \\ \boldsymbol{m}_3^T \end{bmatrix}$$

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \times \begin{bmatrix} \boldsymbol{m}_1^T \boldsymbol{X}_i \\ \boldsymbol{m}_2^T \boldsymbol{X}_i \\ \boldsymbol{m}_3^T \boldsymbol{X}_i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

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 destection

- 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)

Stereo & Camera Calibration

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:
$p_i = \lambda M X_i, \lambda \neq 0 \rightarrow p_i \times M X_i = 0$
Cross product → skew symmetric

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

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

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

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

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

$$G = H \otimes F$$

$$\boldsymbol{min}\ ||\boldsymbol{Lh}||^2$$
$$\boldsymbol{s.t.}\ \|\boldsymbol{h}\|^2 = \boldsymbol{1}$$

- h are unknowns (h1 is row)
- equations: $x' = \frac{h_1^t p}{h_3^t p}$
- minimize: $x' -> h_1^t p - x'(h_3^t p) = 0$

$$L = [\ldots\ldots\ldots\ldots\ldots\ldots \\ p_i^T \quad 0 \quad -x'_i p_i^T \\ 0 \quad p_i^T \quad -y'_i p_i^T \\ \ldots\ldots\ldots\ldots\ldots\ldots]$$

- L =
- $min(h^t A h)$ s.t. $||h||^2 = 1$
  - smallest eigenvalue solves

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

- 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

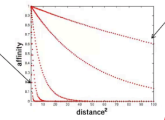
- minimize SSD between all points in cluster

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)

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

Reducing noise = average across img.
Correlation Filtering: (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)

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Blur with Box **S**harpening Filter **S**hift Left by 1 pi

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

<- **S**harpens 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):

Grassman's Law

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)

Linear Color Space

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)

Non-Linear Color Space

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)

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{v}$$

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

x_1, y_1

$$\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{v}\|^2 = \left\|\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} - \begin{bmatrix} mx_1 + b \\ mx_2 + b \end{bmatrix}\right\|^2$$
$$= (y_1 - (mx_1 + b))^2 + (y_2 - (mx_2 + b))^2$$

Roberts

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

Prewitt

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

Sobel

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$\frac{\partial f(x,y)}{\partial x} \quad \begin{bmatrix} -1 & 1 \end{bmatrix} \quad \text{Partial Deriv} \quad \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad \frac{\partial f(x,y)}{\partial y}$$

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