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

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

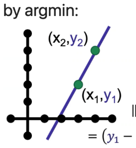
- mirror about y

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

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

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

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



Start with two points (x_i, y_i)

$$y = Av$$

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

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

  - satisfying:

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

or

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

  - 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

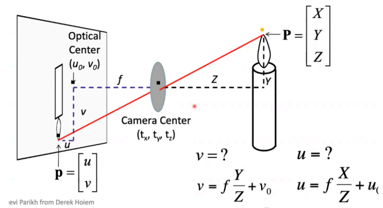
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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

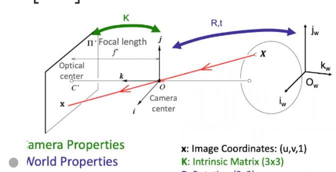


evl Parikh from Derek Hoiem

$v = ?$ $\quad u = ?$

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

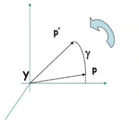
$$K[Rt]X$$



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

Rotation around the coordinate axes, counter-clockwise:

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



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

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

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

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \overbrace{\begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}}^{5} \overbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}}^{6} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

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
    - $\hat{p}'^T E\hat{p} = 0$ $\quad E = [t_x]R$
  - 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: $p'^T Fp = 0$ 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**:

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

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

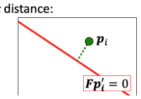
$$\arg\min_{\|f\|=1} \|Uf\|_2^2 \longrightarrow$$ Eigenvector of $U^T U$ with smallest eigenvalue

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

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

May want to minimize *geometric* distance:

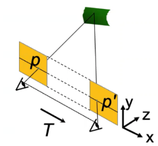
$$\sum_i d(p_i, Fp'_i)^2 + d(p'_i, F^T p_i)^2$$



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



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

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

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

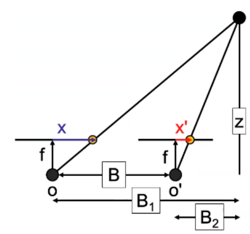
$$[u' \; v' \; 1] \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0 \longrightarrow [u' \; v' \; 1] \begin{bmatrix} 0 \\ -T \\ Tv \end{bmatrix} = 0 \longrightarrow \begin{matrix} -Tv' + Tv = 0 \\ Tv = Tv' \end{matrix}$$

y points are the same

Stereo image rectificiation

- 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

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

$$x - x' = \frac{fB}{z}$$

Disparity

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

- 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

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)

Line fitting

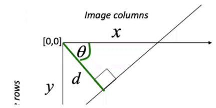
- why fit lines: many objects characterized by presence of straight lines
- problem: extra edge points clutter it, missing lines, noise in edge points

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

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

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

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

- 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
```

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

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

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