Weeks 11-12 Segmentation

segmentation : vital first step

* How to assign pixels to a set of related pixels?

- Edge/boundary : look for differences
- Region : look for similarities

qualities of image

(1) color
(2) texture
(3) motion
*(4) shape
*(5) proximity
*(6) affordance

A case study : robot motion

12\2

affordance

something a physical feature offers in terms of action

handle : human can hold it

roof : shelter from ...

flat surface : can walk

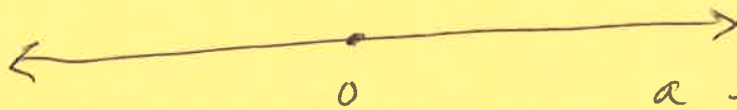
how can image be converted to $\boxed{\text{affordance info}}$?

12|3

The Notion of Symmetry

Given a point set, some operation on the set results in the same set.

Example:



a line ($\mathbb{R}$)

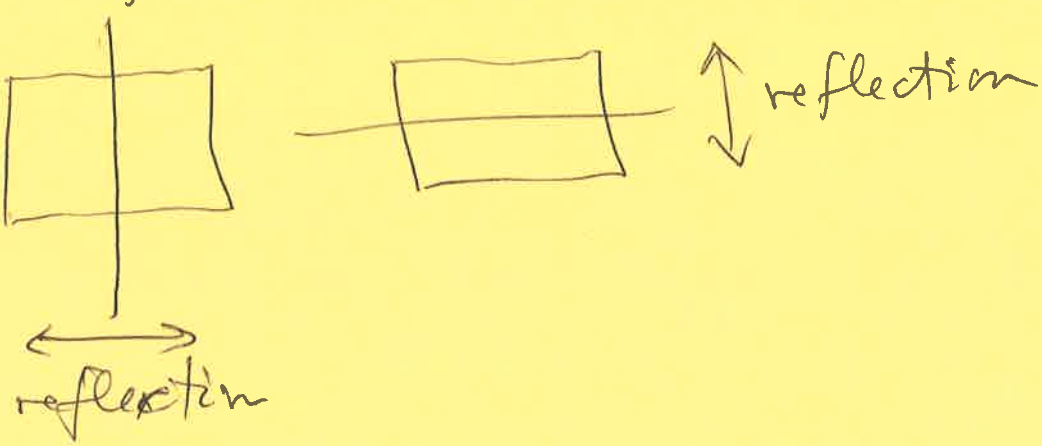
operation: slide the line to the right 1 unit
$(\ell_2(x) = \ell_1(x) + 1 = x + 1$

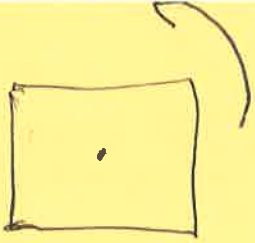
still $\mathbb{R}$

combine action + perception in representation

Consider a square:

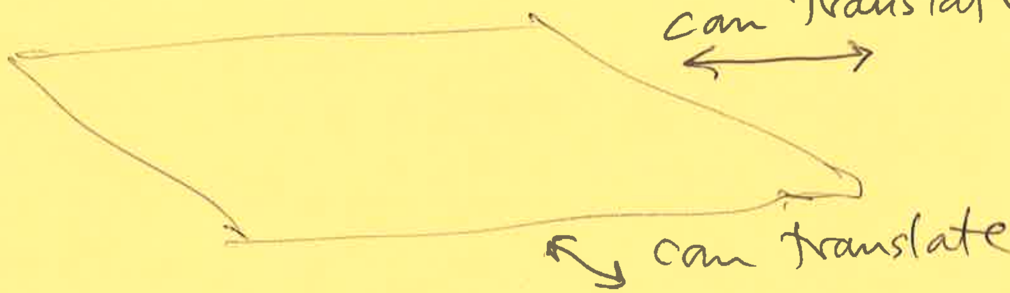
some symmetries:





1/2/4

Consider a plane



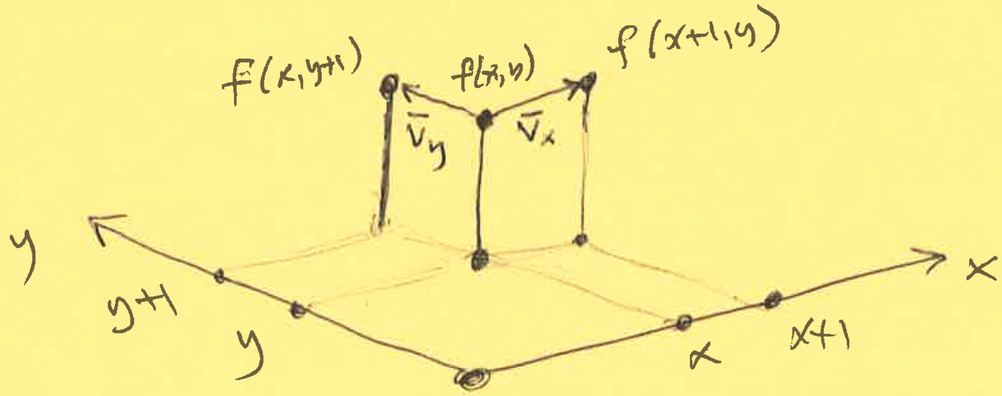
Consider affordance to **move forward on flat surface**

Assume a range map $f(x,y)$ = distance to surface
(called a Monge patch)

Surface normals can be found:

$$\frac{\partial f(x,y)}{\partial x} = \frac{f(x+1,y)-f(x,y)}{\Delta x} = f_x(x,y)$$

$$\frac{\partial f(x,y)}{\partial y} = \frac{f(x,y+1)-f(x,y)}{\Delta y} = f_y(\alpha,y)$$



let $\bar{v}_x = [x+1, y, f(x+1,y)] - [x, y, f(x,y)] = [1, 0, f_x(x,y)]$

$\bar{v}_y = [x, y+1, f(x,y+1)] - [x, y, f(\alpha,y)] = [0, 1, f_y(x,y)]$

normal $\bar{n} = \bar{V}_x \times \bar{V}_y$

$$= \begin{vmatrix} 0 & f_x(x,y) \\ 1 & f_y(x,y) \end{vmatrix} \bar{i} - \begin{vmatrix} 1 & f_x(x,y) \\ 0 & f_y(x,y) \end{vmatrix} \bar{j} + \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} \bar{k}$$

$$= -f_x(x,y)\,\bar{i} - f_y(x,y)\,\bar{j} + 1\,\bar{k}$$

See Powerpoint

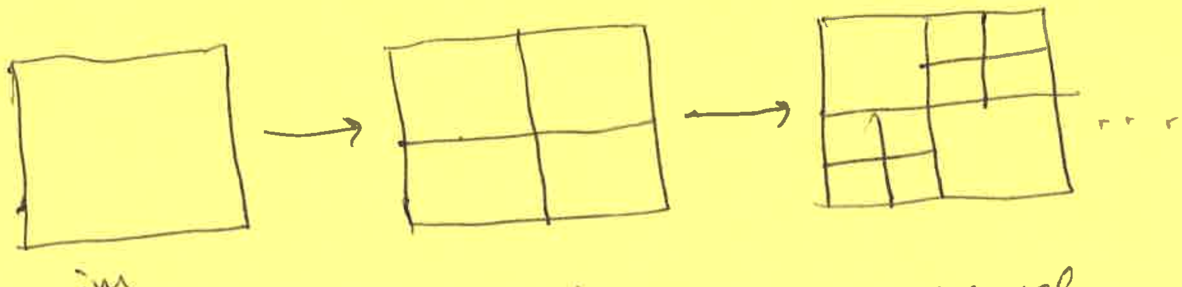
Region growing + splitting

group similar pixels : connected
- * to seed value
- * to average value of region
- * smoothness

separate dissimilar pixels

example 10.2 shows using quadtrees

quadtree: split image recursively into 4 pieces



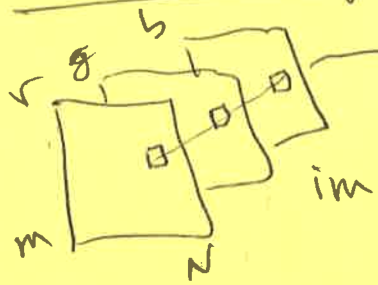
* if region is not homogeneous enough
  split

* after splitting, merge neighboring regions that are similar.

* do this until no change.

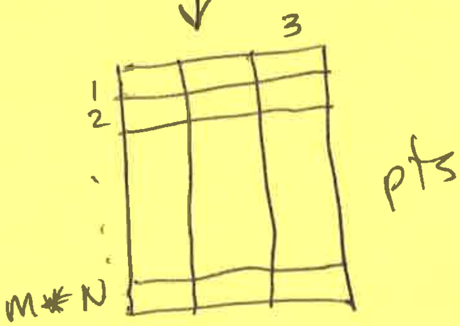
12/7

Other techniques: Kmeans, spectral graph



→ $M*N$ vectors (here triples)

↓ reshape(im, M*N, 3)



↓ Kmeans

[cidx, ctrs] = kmeans(pts, k)

CS4640-week11

spectral graph

given $M \times N \times P$ image,

form similarity matrix:

$S(p_1, p_2)$ = similarity of pixel $p1$ to pixel $p2$

e.g., $S(p_1, p_2) = \text{norm}([p_{11}, p_{12}\ p_{13}] - [p_{21}, p_{22}, p_{23}])$

Find eigenvalues and eigenvectors of S:

[V, D] = eig(S);   % full eigenvalues

the eigenvector associated with the largest eigenvalue separates pixels into similarity classes.

see CS4640-week 11

Differences: edge detection

Revisit Canny edge detector

1. Smooth image with Gaussian kernel
2. Find edge strength, e.g., $E(x,y) = \underbrace{|G_x(x,y)| + |G_y(x,y)|}_{\text{sobel}}$
3. Get edge direction $\theta(x,y) = \tan^{-1} \frac{G_y(x,y)}{G_x(x,y)}$
4. Digitize edge direction
5. Non-maxima suppression: thins and localizes
6. Hysteresis: use 2 thresholds: T_1 & T_2
   - if $|E(x,y)| < T_1$ reject
   - if $|E(x,y)| > T_2$ accept
   - if $T_1 < |E(x,y)| < T_2$ reject unless connected by edge pixel path to $> T_2$ edge

example in text

12/9

Interest operators

points that differ in significant way from neighborhood

$$\nabla I = I(x+\nabla x, y+\nabla y) - I(x,y) = \nabla x f_x + \nabla y f_y$$

where ∇I : change in intensity

$(\nabla x, \nabla y)$: small offset — usually $\Delta x, \Delta y$

$[f_x, f_y]$: gradient at (x,y)

---

consider function of t: $F(t)$ for t near t_0

$F(t_0+\Delta t) \approx F(t_0)$

$F(t_0+\Delta t) \approx F(t_0) + F'(t_0)\Delta t$

⋮

Taylor series terms ↓

To approximate $f(x_0+\Delta x, y_0+\Delta y)$:

$f(x_0+\Delta x, y_0+\Delta y) = F(1)$

$f(x_0+t\Delta x, y_0+t\Delta y) = F(t)$

$x_0, \Delta x, y_0, \Delta y$ are constants

$F(t) = f(x(t), y(t))$ $\quad x(t) = x_0 + t\Delta x \quad y(t) = y_0 + t\Delta y$

$\frac{d}{dt}x(t) = \Delta x \qquad \frac{dy(t)}{dt} = \Delta y$

$$\frac{dF}{dt}(t) = \frac{\partial f}{\partial x}(x(t), y(t))\frac{d}{dt}x(t) + \frac{\partial f}{\partial y}(x(t), y(t))\frac{d}{dt}y(t)$$

$$= f_x(x(t), y(t))\Delta x + f_y(x(t), y(t))\Delta y$$

point is interesting when:

$(\nabla I)^2$ is large for any direction

$$(\nabla I)^2 = (\nabla x \ \nabla y)\begin{bmatrix} \sum_\Omega f_x^2 & \sum_\Omega f_x f_y \\ \sum_\Omega f_x f_y & \sum_\Omega f_y^2 \end{bmatrix}\begin{bmatrix} \nabla x \\ \nabla y \end{bmatrix}$$

$$\bar{V}^T \qquad F \qquad \bar{V}$$

eigenvalues of F ~~give~~ minor + major axis lengths of an ellipse

criteria: (Haralick)

(1) $\frac{\lambda_1 + \lambda_2}{2}$ large — radius high in all directions

(2) $1 - \left(\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}\right)^2$ not too small — axis lengths similar

---

criterion: (Harris)

$$R(x,y) = (1-2k)\sum_\Omega f_x^2 \sum_\Omega f_y^2 - k\left[\left(\sum_\Omega f_x^2\right)^2 + \left(\sum_\Omega f_y^2\right)^2\right] - (f_x f_y)^2$$

$k = 0.04$

12/11

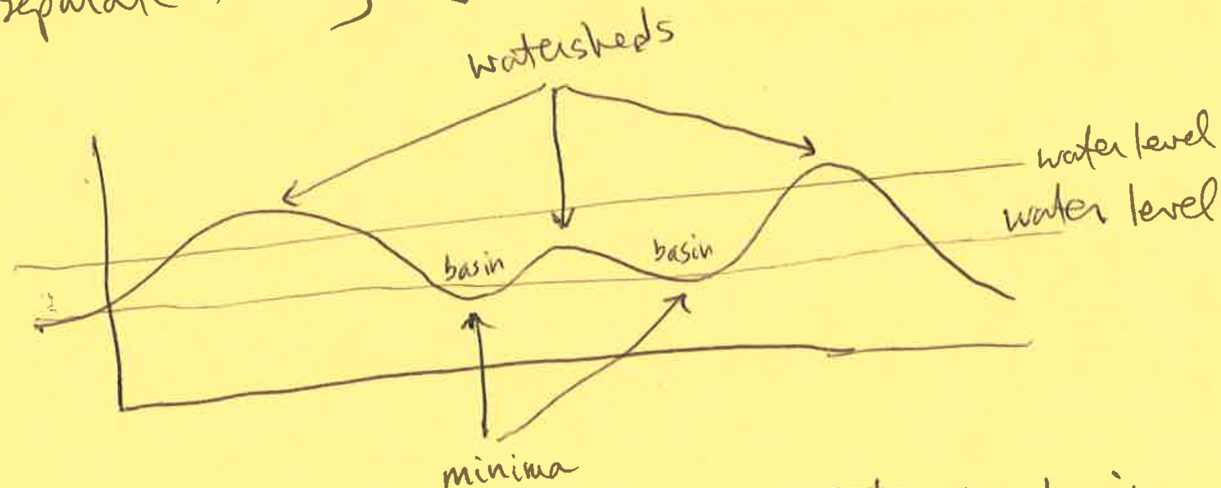
Interest point method

1. Smooth image with Gaussian
2. ∀ pixel, compute gradient
3. —
4. ∀ pixel, compute R
5. Choose local maxima of R as interest points

see Example 10.4 in text

Watershed segmentation

separate touching objects



rain goes to lowest point : accumulates in basin

places where rain can go either way : watershed

Algorithm

On input: binary image   on output: L label image

D = bwdist(~im);   see text + Matlab help

D = -D

D(~im) = -Inf;   L ← initialize to 0

Vals ← unique values in D that are > -Inf

starting at lowest val in vals

set label image L to L + pixels at next value which don't merge 2 previous labeled components

until no more vals

* when adding in next level, don't add to L pixels that merge connected components but keep track of them and set them all to, say -1, after done.

11/7

Consider 3D example

CS4640_Week 10

Given N observations on M variables

i.e., N M-tuples

(zero mean set)

$\bar{x}_i$

define $y_{ij} = \sum_{j=1}^{M} a_{ij} x_{ij}$

$$\underbrace{\begin{bmatrix} y_{i1} \\ y_{i2} \\ \vdots \\ y_{im} \end{bmatrix}}_{\bar{y}_i} = \underbrace{\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ \vdots & & & \\ a_{m1} & & & a_{mm} \end{bmatrix}}_{R} \underbrace{\begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{im} \end{bmatrix}}_{\bar{x}_i}$$

$\bar{y}_i = R\bar{x}_i$

$C_y = \bar{y}\bar{y}^T = R\bar{x}\bar{x}^T R^T = R C_x R^T$

eigenvalue problem

pts → 0-mean pts → covariance matrix

→ V, D

see CS4640_Week 1

11/8

PCA for modeling data

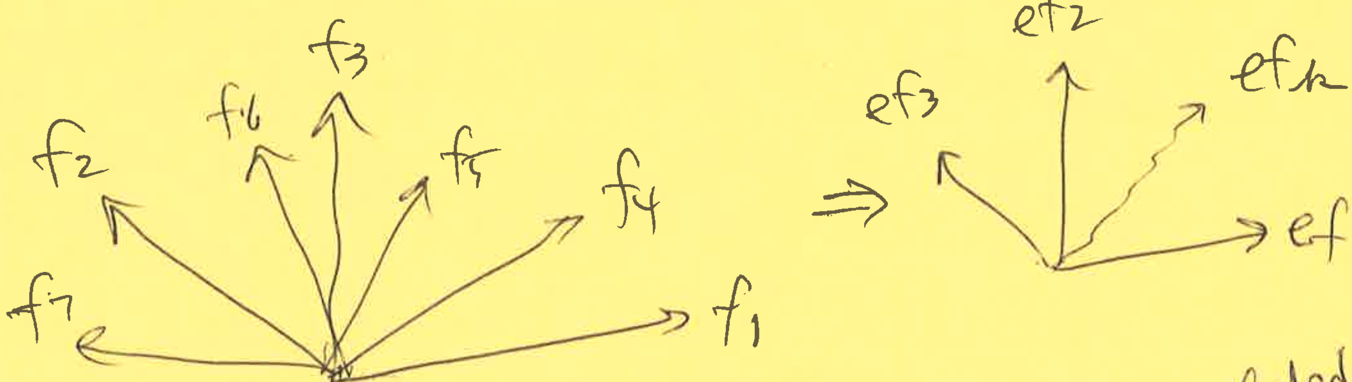
e.g., human faces

given a DB of faces,

determine a set of eigenfaces

and represent new faces in terms of

the coordinate value of the PCA transform



k decorrelated dims

New face is projected onto each

ef axis & the coordinates used

to represent the face

$$a_i = \frac{P_i^T (I - \bar{I})}{\lambda_i} \approx \frac{\text{eigenvec (face - mean)}}{\text{eigen val}}$$