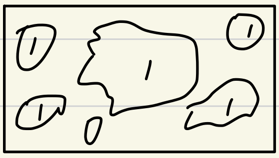
# Intro to segmentation

1) basic region growing

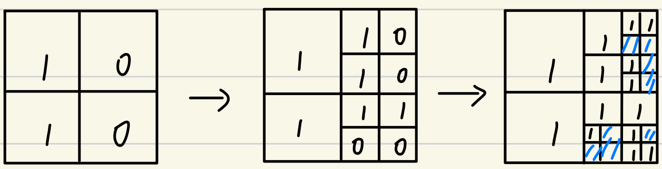


if we only want "common" pixels near one point

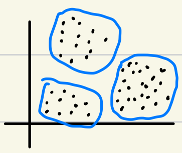
basic algo:
1) from input image I(x,y) get a binary "seed" image S(x,y) for location of intensity (e.g by thresholding)
2) reduce seed connected components down to single point each.
3) let T(x,y) = 1 if I(x,y) satisfies some predicate/ condition and 0 else
e.g (x,y) is 8-connected to seed point $(x_i, y_i)$
and $|I(x,y) - I(x_i, y_i)| \leq T$

matlab: grayconnected

2) region split and merge
specify a condition/rule



3) clustering and superpixels



idea of k-means clustering: gather N-dimensional data into natural cluster (user choose number of clusters)

algo:
1) specify an initial set of cluster centers
$m_1 \cdots m_k \in \mathbb{R}^n$
2) for each of $x_i \in \mathbb{R}^n$ in dataset, assign it to closest cluster

$x_i \in$ cluster $j$ if $\|x_i - m_j\| \leq \|x_i - m_k\|$ for $k \neq j$

3) update the means $m_j$ = average values of all x in cluster j

$$= \frac{1}{|C_j|} \sum x \quad , j = 1 \cdots k$$

4) keep alternating 2) and 3) until $m_j$ stop changing

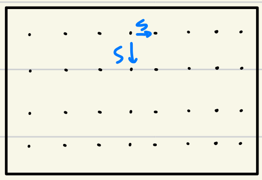
General, initialize k-mean with k random locations, converge.
Do this N times, take best result

Modification of k-means used in image processing: superpixels
Region of image that are contiguous and have similar intensity/color

why do this?
- more compact (e.g thousand of super-pixels could represent millions of pixels)
- "keeps things together" better for subsequent segmentation; computationally efficient.

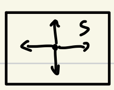
how do we select:
slic super-pixels - simple linear iterative clustering
idea: cluster 5-D vector $[\underbrace{r,g,b}_{\text{colour}},\underbrace{x,y}_{\text{location}}]$

1) initialize superpixel centers by sampling N locations on a regular grid in image plane



move slightly w/in 3*3 neighbourhood to lie on lowest gradient position ( dont want to start on an edge)

2) for each cluster center $m_i$ , compute distance (TBD) b/t $m_i$ and each pixel in a neighbourhood $m_i$



assign pixel to cluster i if its distance is better than it was cluster value

3) update cluster center like in k-means
4) repeat until convergence
5) optional: replace colors of pixels in each cluster with average

what is the distance function?
combination of

$$d_c = \left\| \begin{bmatrix} R \\ G \\ B \end{bmatrix}_i - \begin{bmatrix} R \\ G \\ B \end{bmatrix}_j \right\|_2 \text{ color} \qquad \left\| \begin{bmatrix} x \\ y \end{bmatrix}_i - \begin{bmatrix} x \\ y \end{bmatrix}_j \right\|$$

$d_s$ = spatial

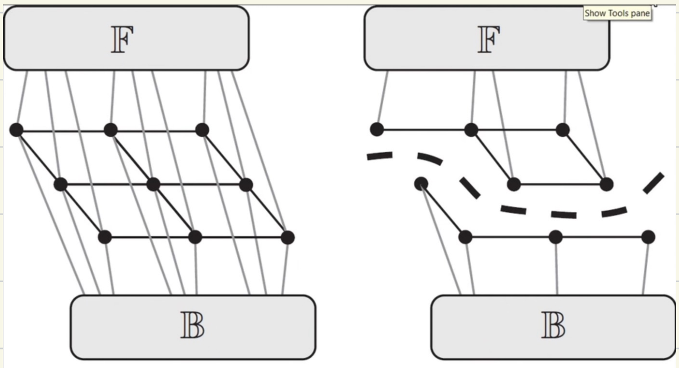
$$D = \sqrt{\left(\frac{d_c}{c}\right)^2 + \left(\frac{d_s}{s}\right)^2}$$

c = max color distance

s = max spatial distance

use <u>c</u> to tune tradeoff,
c big: superpixels more compact
c small: more tightly stuck to image boundary

Graphic cut segmentation
idea: separate foreground and background




a cut is a set of edges that when removed separates F and B
we assign a <u>weight</u> to each edge (both pixel-pixel and pixel-terminal) and we want to find the minimum cut i.e C that minimizes

$$\sum_{(i,j) \in C} w_{ij}$$

between adjacent pixels, we could use:

$$w_{ij} = \frac{1}{dist(i,j)} e^{-\frac{1}{2\sigma^2}\|I_i - I_j\|^2}$$

$I_i \approx I_j$, $e^0 = 1$

$I_i \not\approx I_j$, $e^{-big} = 0$

low cost to cut dissimilar edges
we let the user "scribble" on the image to denotes some initial foreground and background pixels, which from probability distributions

$P_F$(colour) $\qquad$ $P_B$(colour)

1) scribble pix forced to stick with one terminal
e.g FG pixel

$w_{iF} = \infty$, $w_{iB} = 0$

2) other pixels

$$w_{iF} = -\lambda \log F_B(I_i)$$

$$w_{iB} = -\lambda \log F_F(I_i)$$