Computer Vision
for HCI

Image Segmentation

and

Template Matching

1

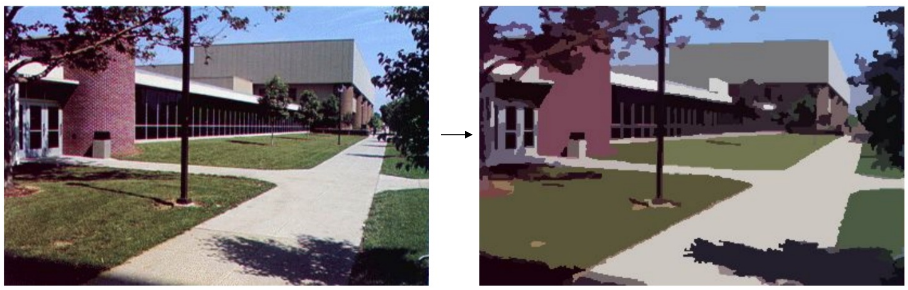
Image Segmentation

- Goal: Partition an image into distinct regions containing each pixels with similar attributes
- Use "*discontinuity*" and/or "*similarity*" approach
  - *Discontinuity:* segment into regions based on discontinuity (gradient or edge detection)
  - *Similarity:* Merge similar regions (clustering, region growing etc.)
- Topics
  - Simple Segmentation
  - Segmentation by Clustering
  - Superpixel Segmentation

2

2

Goal

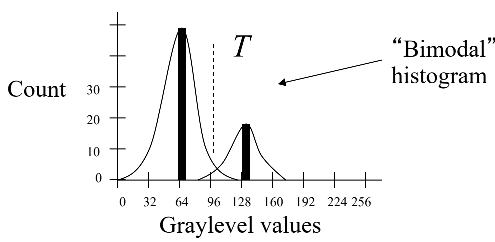


3

3

Recap: Otsu's Simple Segmentation

- Distribution of graylevels can be used to determine binary threshold
- Histogram graphs number of pixels in the image with a particular graylevel, as a function of the possible graylevels
  - Find peaks and set threshold between peaks

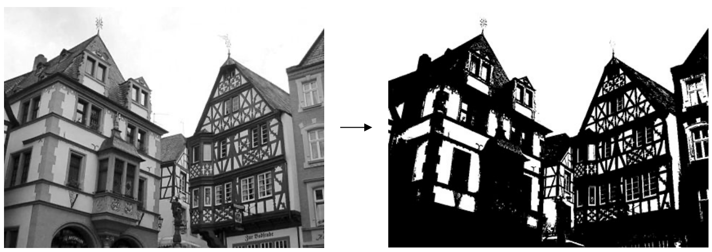


4

4

Otsu's Method

- "A threshold selection method from graylevel histograms", IEEE Trans on Sys., Man, and Cyb., Vol 9, No 1, pp 62-66, 1979.
  - Basic idea: threshold is chosen such that the division in the histogram yields the largest reduction in standard deviation of the pixel intensities (black, white)
  - Matlab: graythresh()



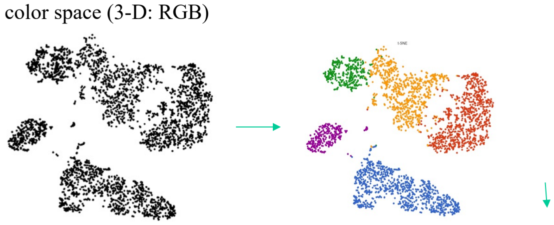
5

5

Image Segmentation by Clustering

Identify groups of pixels that "go together"

Each "point" is a pixel in color space (3-D: RGB)



- K-Means
- Mean-Shift Clustering

6

6

K-Means

7

7

K-means Clustering

- Each "point" is a 3-D vector of color (RGB)
- Initialization:
  - Choose *k* cluster center points (how pick *k*?)
- Repeat:
  - **Assignment step:**
    - For every point, find its closest center point
  - **Update step:**
    - Update every center point as the mean of its assigned points
- Until:
  - The maximum number of iterations is reached, or
  - No changes during the assignment step, or
  - The average distortion per point drops very little

8

8

K-means: Initialization

- K-means is *extremely sensitive* to initialization
- Bad initialization can lead to
  - Poor convergence speed
  - Poor overall clustering
- How to initialize?
  - Randomly from data
  - Try to find K "spread-out" points (K-means++)
- Try multiple initializations and pick best result
  - Minimize total "distortion/inertia" (sum of distances of points to their cluster centers)

Data Cluster center

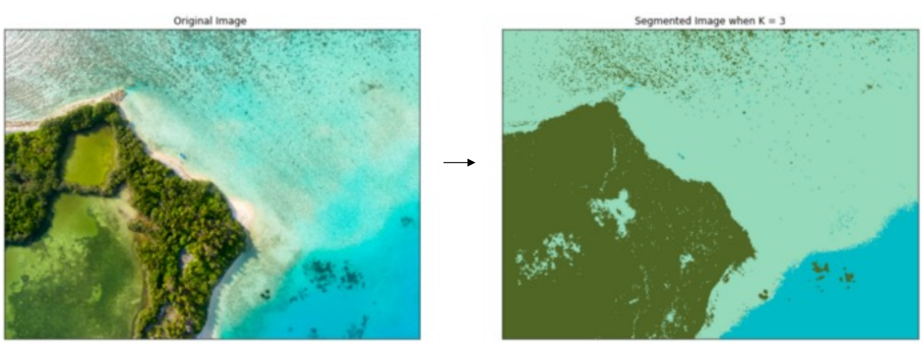
$$J(\mu, r) = \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_{nk} ||x_n - \mu_k||^2$$

Whether x_j is assigned to μ_i

9

9

Example



10

10

Mean-Shift

11

11

Mean-Shift segmentation

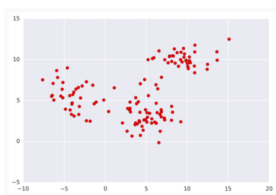
- Recall Mean-Shift tracking lecture…
- Used here for clustering
  - Unlike K-means, do **not** need initial 'K' number
- Assigns the data points to clusters iteratively by shifting points towards the "local modes"
  - Mode: The highest density of data points in the region, in the context of the Mean-Shift

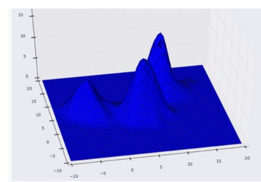
12

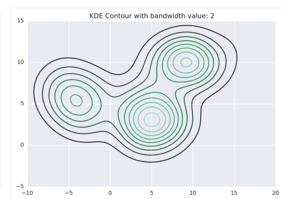
12

Mean-Shift clustering

- **Related to "Kernel Density Estimation" (KDE)**
  - Imagine the data is sampled from a probability distribution
  - Goal: Estimate the underlying distribution (also called the probability density function) for a set of data
    - Place kernel (think "weighting function") on each point
    - Adding all the individual kernels generates a probability surface (e.g., density function)





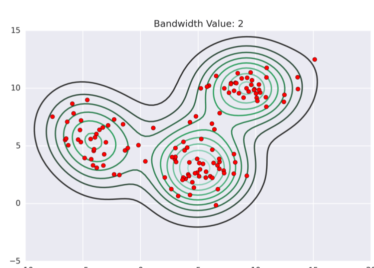


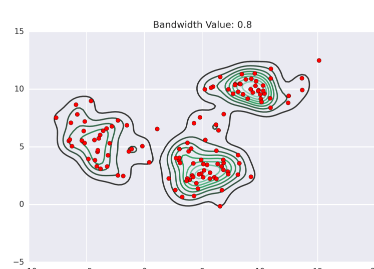
13

13

Mean-Shift Clustering

- Idea:
  - Make points climb up the hill to the nearest peak on the density surface
  - *Iteratively* shift each point "uphill" until it reaches a peak





14

14

Mean-Shift Algorithm

- Define datapoint *x*
  - Color only: [*R, G, B*]
  - Spatial and Color: [*x-loc, y-loc, R, G, B*]
- For each datapoint *x*, find the neighboring points *N(x)* of *x,* given Kernel function/window *K*
- For each *x*, calculate the ***mean shift*** *m(x)*:

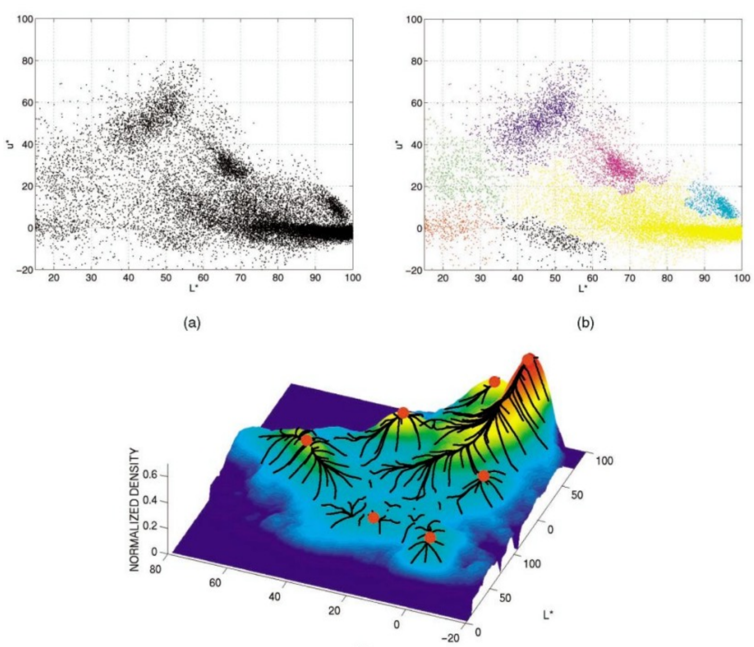
$$m(x) = \frac{\sum_{x_i \in N(x)} K(x_i - x) x_i}{\sum_{x_i \in N(x)} K(x_i - x)}$$

- Then update each *x* with *x* ← *m(x)*
- Repeat *n times,* or until the points stabilize

More details (Matt Nedrich):
https://spin.atomicobject.com/2015/05/26/mean-shift-clustering/

15

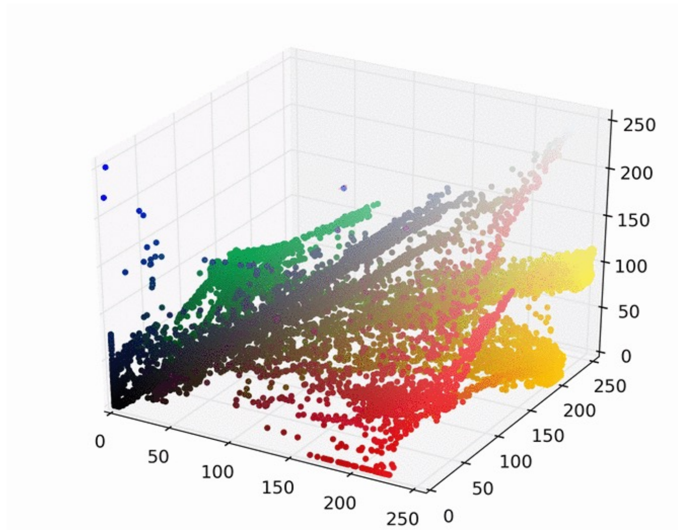
15



16

16

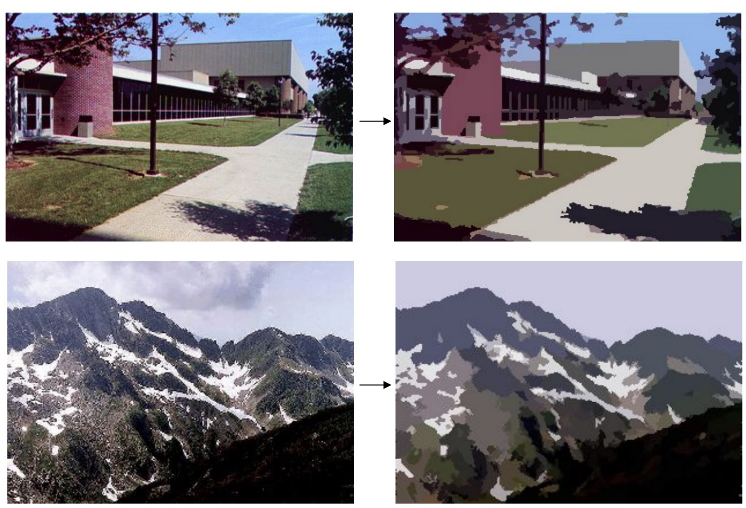
RGB Visualization



17

17

Examples



18

18

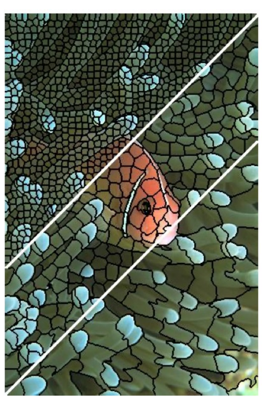
Superpixels

19

19

Superpixel Segmentation

- "Superpixels" capture local visual redundancy in the image
  - **SLIC** Superpixel algorithm



20

20

SLIC Superpixel Segmentation

- Generated by clustering pixels based on:
  - Color similarity, and
  - Spatial proximity in the image
- Employ 5-D vector per pixel: [L, A, B, x, y]
  - [L, A, B] = pixel color vector in CIELAB color space
    - (L): intensity, (A, B): color
    - "Perceptual color space" with Euclidean distance properties
  - x, y = pixel position (or col, row)

21

SLIC Superpixel Algorithm

- Initially choose K = number of desired superpixels
- Divide image into regular "grid" steps S (for the desired K)

Algorithm 1 Efficient superpixel segmentation

1: Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.
2: Perturb cluster centers in an $n \times n$ neighborhood, to the lowest gradient position.
(to the pixel with smallest gradient magnitude in 3x3 region)
3: **repeat**
4: **for** each cluster center C_k **do**
5: Assign the best matching pixels from a $2S \times 2S$ square neighborhood around the cluster center according to the distance measure (see next few slides)
6: **end for**
7: Compute new cluster centers and residual error E $\{L1$ distance between previous centers and recomputed centers$\}$
8: **until** $E \leq$ threshold

22

Notation

N	Number of pixels in the input image
K	Number of Superpixels used to segment the input image
N/K	Approximate size of each superpixel
$\boldsymbol{S = \sqrt{N/K}}$	For roughly equally sized superpixels there would be a superpixel centre at every grid interval S

23

23

SLIC Superpixel Segmentation

- Distance function between 2 pixels:

$$D_s = d_{lab} + \frac{m}{S} d_{x,y}$$

d_{lab} : LAB distance (Euclidean) between the 2 pixels.

$\frac{1}{S} d_{x,y}$: Euclidean spatial distance, normalized by grid interval S.

m : **compactness** control of a super pixel. Larger values make it more compact.

24

24

Superpixels: More Examples

Approx. 300 / 100 superpixel versions

25

25

Template Matching

26

26

Template Matching Intro

- Want to find areas of a search image that are similar to a given template T

Template T

Search Image

Best Matching Patch in Search Image

General Approaches

- Template-Based:
  - Utilize raw template (pixels) and find best matching patches in search image
    - Sum-of-absolute differences (SAD)
    - Sum-of-squared differences (SSD)
    - Normalized cross-correlation (NCC)

1) Sum-of-Absolute Differences
(SAD)

- Compute **absolute differences of pixel intensities** of template *T* and image patch *P* extracted from search image (note that *P* is the same size as template *T*)

$$SAD(P,T) = \sum_{R,G,B} \sum_{x,y} \left| P(x,y) - T(x,y) \right|$$

- Compute SAD for all patch locations within the search image
- Keep patch with minimum SAD or patches with SAD less than given threshold

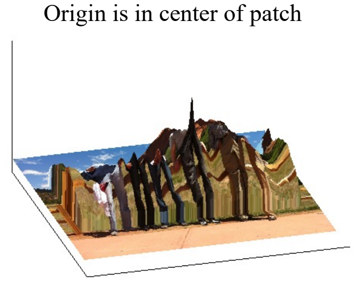
29

SAD Example

Template Image *T*

Search Image

Negative SAD (just for visualization), Origin is in center of patch

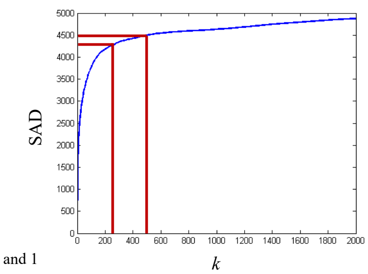


30

SAD Example

k^{th} best matching patch

$k = 1$ $k = 10$ $k = 50$ $k = 100$ $k = 150$ $k = 200$ $k = 250$ $k = 500$ $k = 1000$



Similar???

*Pixel values scaled between 0 and 1

31

2) Sum-of-Squared Differences (SSD)

- Similar to SAD, but replace absolute differences with **squared differences**

$$SSD(P,T) = \sum_{R,G,B} \sum_{x,y} \left(P(x,y) - T(x,y)\right)^2$$

- Compute SSD for all patches within the search image
- Keep patch with <u>minimum</u> SSD

32

SSD Example

Template T

Search Image

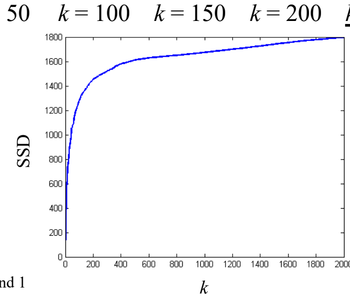
Negative SSD,
Origin is in center of patch

33

SSD Example

k^{th} best matching patch

$k = 1$ | $k = 10$ | $k = 50$ | $k = 100$ | $k = 150$ | $k = 200$ | $k = 250$ | $k = 500$ | $k = 1000$



*Pixel values scaled between 0 and 1

34

Illumination Changes

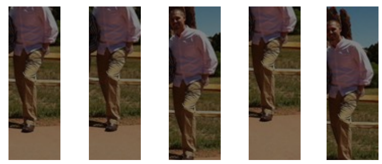
- SAD and SSD can work well if the template and search image have the same brightness
  - Problem: images can have varying illumination conditions

Template T

Search Image

k^{th} best matching patch using SSD

$k = 1$ $k = 10$ $k = 50$ $k = 100$



$k = 150$ $k = 200$ $k = 250$ $k = 500$ $k = 1000$

3) Normalized Cross-Correlation

(NCC)

- Normalize images to remove variations from illumination conditions

Mean of pixel values <u>in patch</u>
(each color computed independently)

$$NCC(P,T) = \sum_{R,G,B} \frac{1}{n-1} \sum_{x,y} \frac{\left(P(x,y) - \overline{P}\right) \cdot \left(T(x,y) - \overline{T}\right)}{\sigma_P \sigma_T}$$

Standard deviation of pixel values in patch
(each color computed independently)

Note: <u>larger</u> values of NCC <u>better</u>!

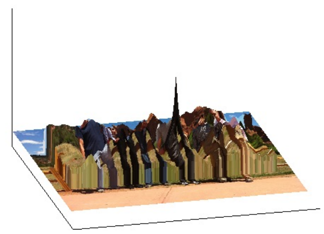
The maximum value is 1 when two <u>1-channel</u> signals are exactly the same: NCC(a, a)=1

NCC Example

Template T

Search Image

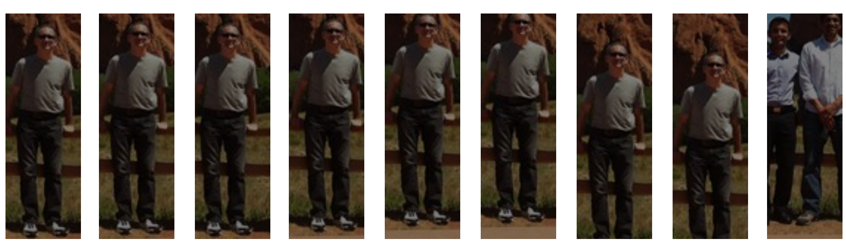
NCC,
Origin is in center of patch



37

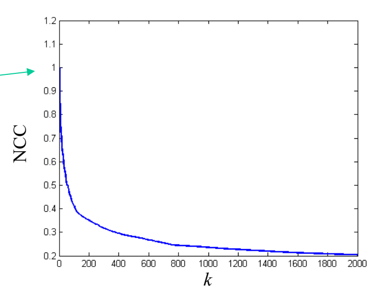
NCC Example

k^{th} best matching patch using **color** images



$k = 1$ $k = 10$ $k = 50$ $k = 100$ $k = 150$ $k = 200$ $k = 250$ $k = 500$ $k = 1000$

NCC divided by 3 in this plot (for RGB)

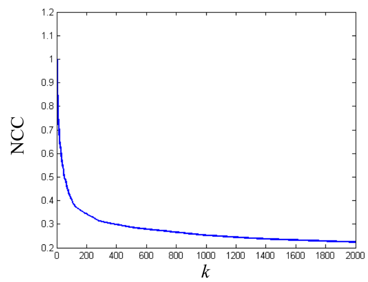


38

NCC Example

k^{th} best matching patch using **grayscale** images

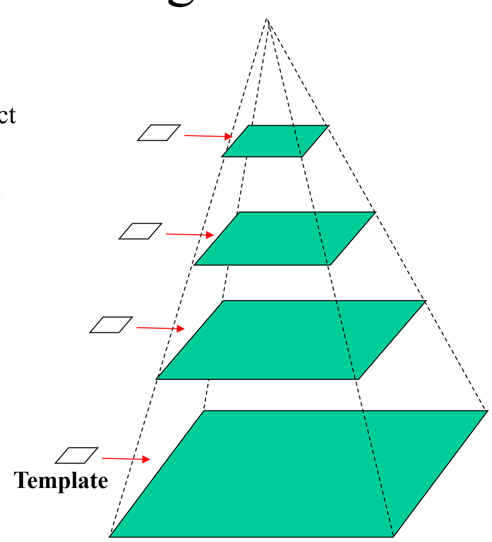
$k = 1$ $k = 10$ $k = 50$ $k = 100$ $k = 150$ $k = 200$ $k = 250$ $k = 500$ $k = 1000$



39

Handling Differing Scales

- Construct **fixed-sized** template of the smallest size you want to detect
- Scan through image pyramid
  - Detects larger scales of object higher in the pyramid
- Efficient scanning method, with less pixels to examine overall
  - Instead of repeatedly scaling up the template and scanning original full-sized image multiple times



Pyramid of search image

40

Summary

- K-Means
  - Choose cluster centers and label every pixel based on its nearest neighbor
  - Minimize total distortion
  - Sensitive to initialization
- Mean-Shift
  - Iteratively shifts data towards peaks
- Superpixel Segmentation
  - Clustering small pixel regions based on color similarity and proximity
- Template Matching
  - SAD, SSD, NCC

41

41