Segmentation & Grouping

Tuesday, Sept 23

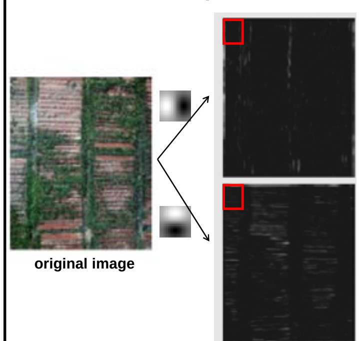
Last time

- Texture is a useful property that is often indicative of materials, appearance cues
- **Texture representations** attempt to summarize repeating patterns of local structure
- **Filter banks** useful to measure redundant variety of structures in local neighborhood
  - Feature spaces can be multi-dimensional

Texture representation

- Textures are made up of repeated local patterns, so:
  - **Find the patterns**
    - Use filters that look like patterns (spots, bars, raw patches…)
    - Consider magnitude of response
  - **Describe their statistics within each local window**
    - Mean, standard deviation
    - Histogram
    - Histogram of “prototypical” feature occurrences

Texture representation: example

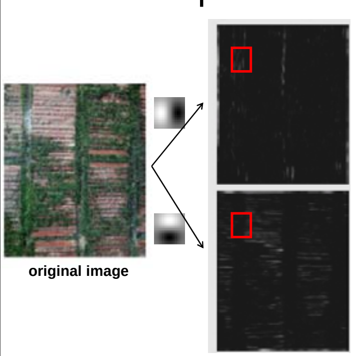


	mean d/dx value	mean d/dy value
Win. #1	4	10

⋮

statistics to summarize patterns in small windows

Texture representation: example

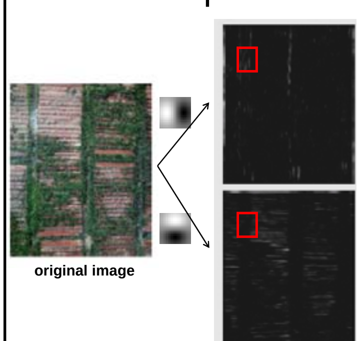


	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2	18	7

⋮

statistics to summarize patterns in small windows

Texture representation: example

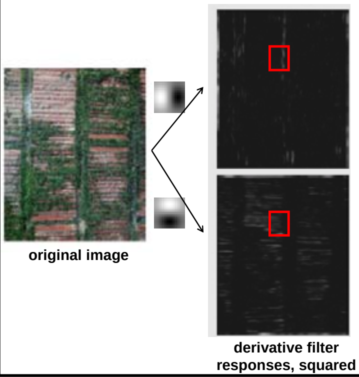


	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2	18	7

⋮

statistics to summarize patterns in small windows

Texture representation: example

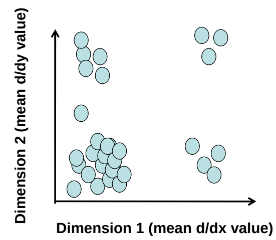


	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2 ⋮	18	7
Win.#9	20	20

⋮

statistics to summarize patterns in small windows

Texture representation: example

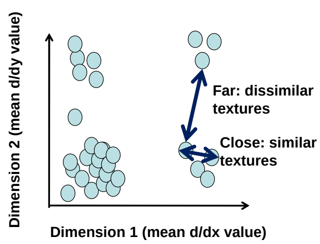


	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2 ⋮	18	7
Win.#9	20	20

⋮

statistics to summarize patterns in small windows

Texture representation: example

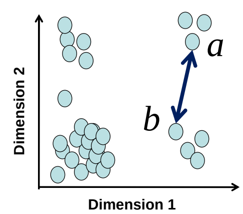


	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2 ⋮	18	7
Win.#9	20	20

⋮

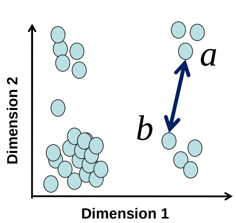
statistics to summarize patterns in small windows

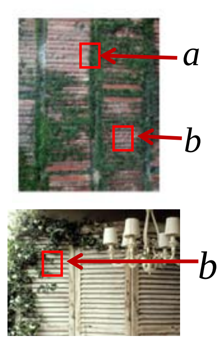
Texture representation: example



$$D(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$
$$= \sqrt{\sum_{i=1}^{2} (a_i - b_i)^2}$$

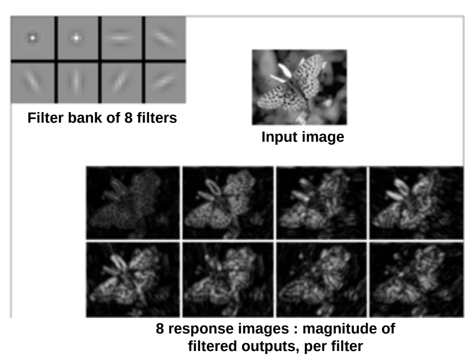
Texture representation: example





Distance reveals how dissimilar texture from window a is from texture in window b.

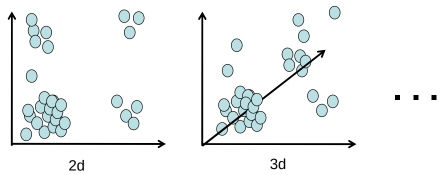
Example application of a filter bank



d-dimensional features

$$D(a,b) = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2}$$

General definition of inter-point distance



Review questions

- When describing texture, why do we collect filter response statistics within a window?
- What is the Markov assumption?
  - And why is it relevant for the texture synthesis technique of Efros & Leung?

Outline

- What are grouping problems in vision?
- Inspiration from human perception
  - Gestalt properties
- Bottom-up segmentation via clustering
  - Algorithms: k-means, graph-based
  - Features: color, texture, …

Grouping in vision

- Goals:
  - Gather features that belong together
  - Obtain an intermediate representation that compactly describes key image (video) parts

Examples of grouping in vision

[Figure by J. Shi]

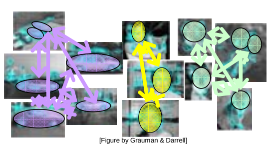
Determine image regions

[http://poseidon.csd.auth.gr/LAB_RESEARCH/Latest/imgs/SpeakDepVidIndex_img2.jpg]

Group video frames into shots

[Figure by Wang & Suter]

Figure-ground



[Figure by Grauman & Darrell]

Object-level grouping

Grouping in vision

- Goals:
  - Gather features that belong together
  - Obtain an intermediate representation that compactly describes key image (video) parts
- Top down vs. bottom up segmentation
  - Top down: pixels belong together because they are from the same object
  - Bottom up: pixels belong together because they look similar
- Hard to measure success
  - What is interesting depends on the app.

What things should be grouped?
What cues indicate groups?

Similarity




http://chicagoist.com/attachments/chicagoist_alicia/GEESE.jpg, http://wwwdelivery.superstock.com/WI/223/1532/PreviewComp/SuperStock_1532R-0831.jpg

Symmetry




http://seedmagazine.com/news/2006/10/beauty_is_in_the_processingtim.php

Common fate

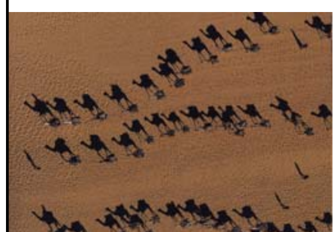
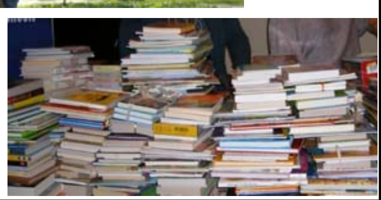


Image credit: Arthus-Bertrand (via F. Durand)

Proximity

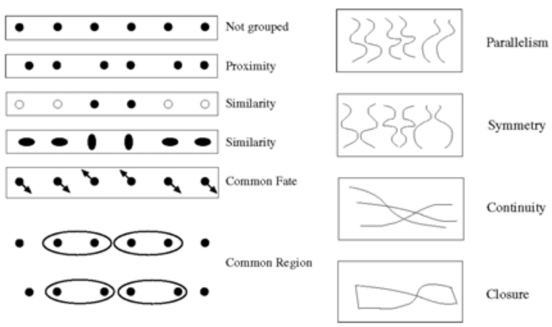



http://www.capital.edu/Resources/Images/outside6_035.jpg

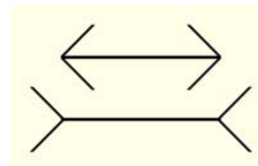
Gestalt

- Gestalt: whole or group
  - Whole is greater than sum of its parts
  - Relationships among parts can yield new properties/features
- Psychologists identified series of factors that predispose set of elements to be grouped (by human visual system)

Some Gestalt factors

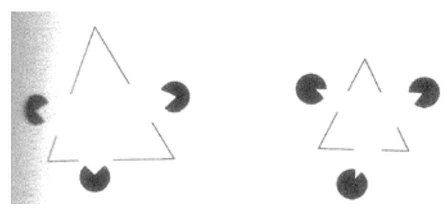


Muller-Lyer illusion



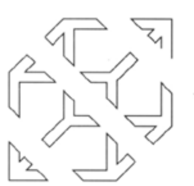
Gestalt principle: grouping key to visual perception.

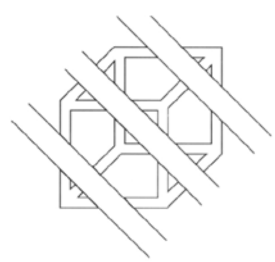
Illusory/subjective contours



Interesting tendency to explain by occlusion

In *Vision*, D. Marr, 1982

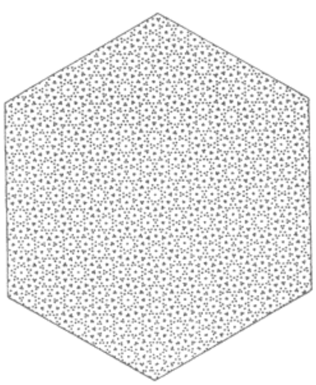




Continuity, explanation by occlusion

D. Forsyth

Figure-ground



In *Vision*, D. Marr, 1982; from J. L. Marroquin, "Human visual perception of structure", 1976.

Gestalt

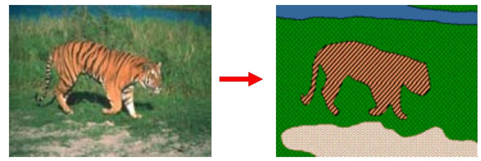
- Gestalt: whole or group
  - Whole is greater than sum of its parts
  - Relationships among parts can yield new properties/features
- Psychologists identified series of factors that predispose set of elements to be grouped (by human visual system)
- **Inspiring observations/explanations, but not necessarily directly useful for algorithms.**

Outline

- What are grouping problems in vision?
- Inspiration from human perception
  – Gestalt properties
- Bottom-up segmentation via clustering
  – Algorithms: k-means, graph-based
  – Features: color, texture, …

Image segmentation

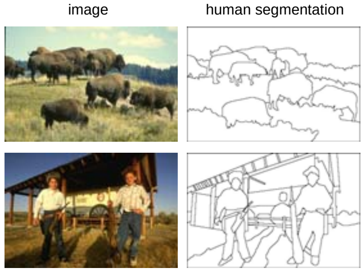
- Goal: identify groups of pixels that go together.



The goals of segmentation

Separate image into coherent “objects”

image / human segmentation



Source: Lana Lazebnik

The goals of segmentation

Separate image into coherent “objects”

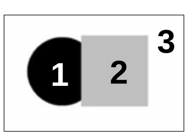
Group together similar-looking pixels for efficiency of further processing

“superpixels”

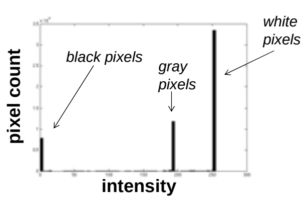
X. Ren and J. Malik. **Learning a classification model for segmentation.** ICCV 2003.

Source: Lana Lazebnik

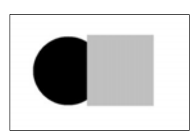
Image segmentation: toy example



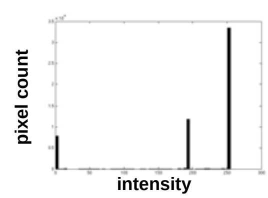
input image

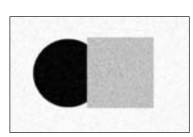


- These intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., *segment* the image based on the intensity feature.
- What if the image isn’t quite so simple?

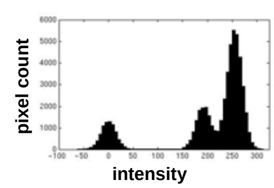


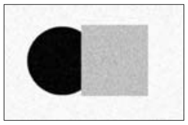
input image



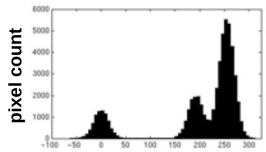


input image

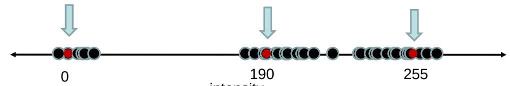


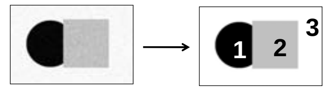


input image



- Now how to determine the three main intensities that define our groups?
- We need to ***cluster.***



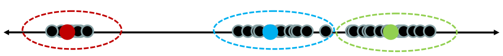


- Goal: choose three "centers" as the representative intensities, and label every pixel according to which of these centers it is nearest to.
- Best cluster centers are those that minimize SSD between all points and their nearest cluster center c_i:

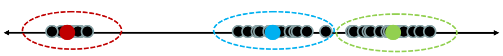
$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

Clustering

- With this objective, it is a "chicken and egg" problem:
  - If we knew the **cluster centers**, we could allocate points to groups by assigning each to its closest center.



  - If we knew the **group memberships**, we could get the centers by computing the mean per group.



K-means clustering

- Basic idea: randomly initialize the *k* cluster centers, and iterate between the two steps we just saw.
  1. Randomly initialize the cluster centers, $c_1, ..., c_K$
  2. Given cluster centers, determine points in each cluster
     - For each point p, find the closest c_i. Put p into cluster i
  3. Given points in each cluster, solve for c_i
     - Set c_i to be the mean of points in cluster i
  4. If c_i have changed, repeat Step 2

Properties
- Will always converge to *some* solution
- Can be a "local minimum"
  - does not always find the global minimum of objective function:

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

Source: Steve Seitz

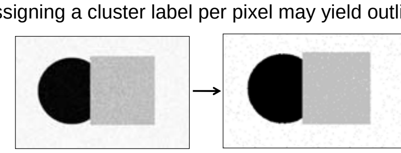
K-means clustering

- Java demo:

http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html

Smoothing out cluster assignments

- Assigning a cluster label per pixel may yield outliers:

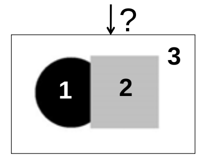


original

labeled by cluster center's intensity

?

- How to ensure they are spatially smooth?

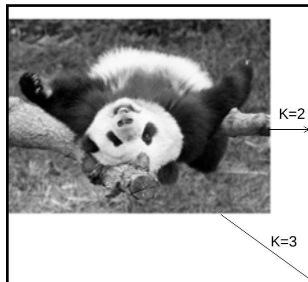


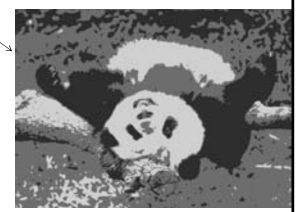
Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity** similarity

Feature space: intensity value (1-d)



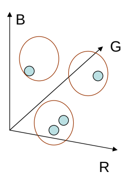


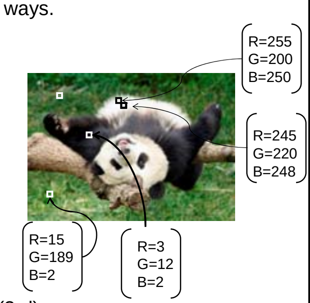
```
img_as_col = double(im(:));
cluster_membs = kmeans(img_as_col, K);

labelim = zeros(size(im));
for i=1:k
   inds = find(cluster_membs==i);
   meanval = mean(img_as_column(inds));
   labelim(inds) = meanval;
end
```

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **color** similarity



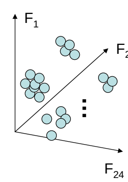


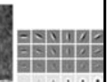
Feature space: color value (3-d)

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **texture** similarity

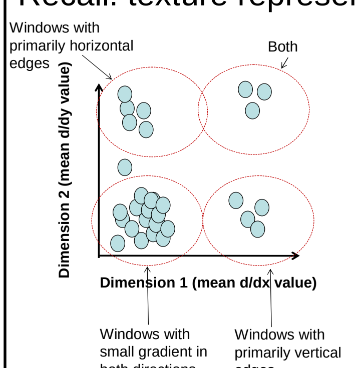




Filter bank of 24 filters

Feature space: filter bank responses (e.g., 24-d)

Recall: texture representation example



	mean d/dx value	**mean d/dy value**
Win. #1	4	10
Win.#2	18	7
⋮		
Win.#9	20	20

⋮

statistics to summarize patterns in small windows

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity** similarity

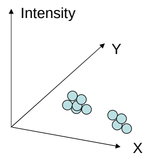
What will the clusters be in this image?

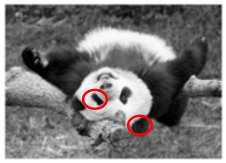
They don't have to be spatially coherent.

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity+position** similarity

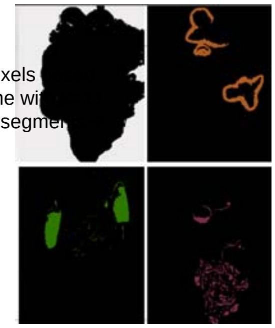




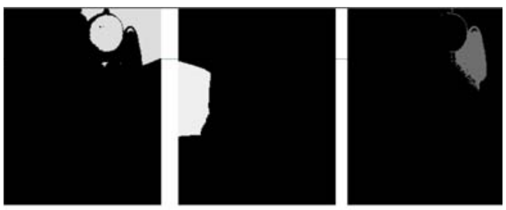
Both regions are black, but if we also include **position (x,y),** then we could group the two into distinct segments; way to encode both similarity & proximity.

Image

Clustering pixels
on color alone wi
yields these segme



Masks showing
four of the clusters



If instead we use both color and position, k-means will yield segments that depend on both.

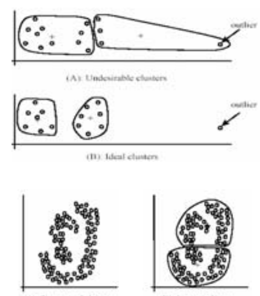
K-means: pros and cons

<u>Pros</u>

- Simple, fast to compute
- Converges to local minimum of within-cluster squared error

<u>Cons/issues</u>

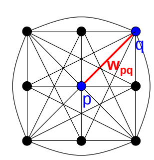
- Setting k?
- Sensitive to initial centers
- Sensitive to outliers
- Detects spherical clusters
- Assuming means can be computed



Outline

- What are grouping problems in vision?
- Inspiration from human perception
  - Gestalt properties
- Bottom-up segmentation via clustering
  - Algorithms: k-means, graph-based
  - Features: color, texture, …

Images as graphs

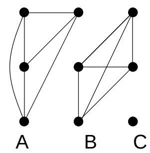


Fully-connected graph

- node (vertex) for every pixel
- link between *every* pair of pixels, **p,q**
- affinity weight w_{pq} for each link (edge)
  - w_{pq} measures *similarity*
    - » similarity is *inversely proportional* to difference (in color and position…)

Source: Steve Seitz

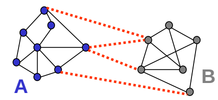
Segmentation by Graph Cuts



Break Graph into Segments
- Delete links that cross between segments
- Easiest to break links that have low similarity (low weight)
  - similar pixels should be in the same segments
  - dissimilar pixels should be in different segments

Source: Steve Seitz

Cuts in a graph: Min cut



Link Cut
- set of links whose removal makes a graph disconnected
- cost of a cut: $$cut(A,B) = \sum_{p \in A, q \in B} w_{p,q}$$

Find minimum cut
- gives you a segmentation
- fast algorithms exist for doing this

Source: Steve Seitz

Minimum cut

- Problem with minimum cut:
  Weight of cut proportional to number of edges in the cut; tends to produce small, isolated components.

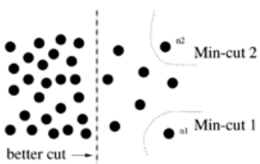
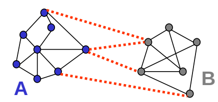


Fig. 1. A case where minimum cut gives a bad partition.

[Shi & Malik, 2000 PAMI]

Cuts in a graph: Normalized cut



Normalized Cut
- fix bias of Min Cut by **normalizing** for size of segments:

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

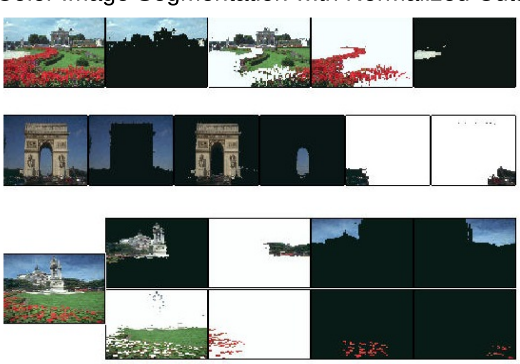
assoc(A) = sum of weights of all edges that touch A

- Approximate solution for minimizing the ncut value : generalized eigenvalue problem.

J. Shi and J. Malik, Normalized Cuts and Image Segmentation, CVPR, 1997

Source: Steve Seitz

Color Image Segmentation with Normalized Cuts



J. Shi and J. Malik, Normalized Cuts and Image Segmentation, CVPR, 1997

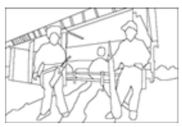
Normalized cuts: pros and cons

Pros:
- Generic framework, flexible to choice of function that computes weights ("affinities") between nodes
- Does not require model of the data distribution

Cons:
- Time complexity can be high
  - Dense, highly connected graphs → many affinity computations
  - Solving eigenvalue problem
- Preference for balanced partitions

Segmentation: Caveats

- We've looked at *bottom-up* ways to segment an image into regions, yet finding meaningful segments is intertwined with the recognition problem.
- Often want to avoid making hard decisions too soon
- Difficult to evaluate; when is a segmentation successful?



Generic clustering

- We have focused on ways to group pixels into image segments based on their appearance
  - Find groups; "quantize" feature space
- In general, we can use clustering techniques to find groups of similar "tokens", provided we know how to compare the tokens.
  - E.g., segment an image into the types of motions present
  - E.g., segment a video into the types of scenes (shots) present

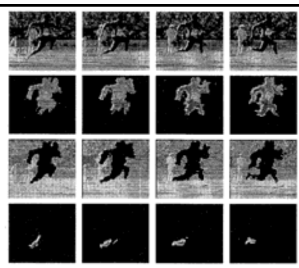
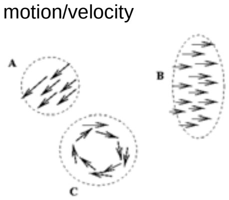


Figure 5: The first row shows an image sequence of Carl Lewis running. Notice that the background is moving to the left as the camera is panning to keep the runner in the center of the image, and therefore background subtraction would not work as an image segmentation technique. The original image size is 200 × 190, and image patches of size 3 × 3 is used to construct the partition graph. Each of the image patches are connected to others that are less than 5 superpixels and 3 image frames away. Row 2 to 4 show the motion segmentation produced by our algorithm. Note these regions found corresponds the runner in row 2, moving background in row 3, and the left lower leg in row 4. The left lower leg is segmented from the runner because it undergoes significant upward rotation in these seven image frames. By recursive cuts and by lowering the maximum allowed *Ncut* value, the other moving limbs can be found.

What if we segment an image into groups of *motions*?

Features = measure of motion/velocity



(We'll look at how to measure motion later in the course.)

Motion Segmentation and Tracking Using Normalized Cuts [Shi & Malik 1998]

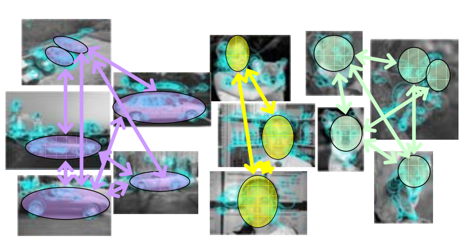
Shot detection:

Segment a video into groups of consecutive frames with similar color distributions

[http://poseidon.csd.auth.gr/LAB_RESEARCH/Latest/imgs/SpeakDepVidIndex_img2.jpg]

Unsupervised object category discovery:

Build a graph of images, with edges weighted by some feature matching score. Partition with graph cuts.



K. Grauman & T. Darrell, Unsupervised Learning of Categories from Sets of Partially Matching Image Features, CVPR 2006.

Next

- Fitting
- Read F&P Chapter 15.1: Hough Transform