Content-Based Image Retrieval (CBIR)

<motivate with bird call project>

Outline

- Problem statement
- Query by example
- Framework overview
- Image features
- Distance/similarity measures
- Evaluation metrics
- Advanced topics (relevance feedback, etc.)

Problem statement

Retrieve images based on their visual content
- Do not use/require manual annotation (text labels)

How to perform this search

1) By keyword.

Example "sunflower"
Advantages: Intuitive for user.
Disad: Requires that images have been annotated. This is difficult to do automatically and requires image understanding.

2) By similarity search.

Given an example image, retrieve images that are visually similar.
→ Query-by-example (QBE)
Ad: Only requires a method to compute visual similarity.
Disad: Requires users to identify query image.
Visual similarity might not correspond to semantic similarity.
Ex: An image of the sun and an image of a tennis ball are visually similar but are semantically very different.

QBE

How to specify query image?

1) Have user "sketch" it.

Ad: Intuitive
Disad: Difficult to compute visual similarity between real and sketched images. Images can be complex and thus difficult to sketch.

2) Use a real image as the query.

Ad: Only requires image-to-image visual similarity.
Disad: Requires user find example image.

We will use second method, query by a real example image.

Framework

Given a query image Q and a set of target images T_i, $i = 1, \ldots, n$, retrieve the images from T_i that are visually similar to Q.

If we had a function f that took as input two images and determined whether the images are similar or not, we could be done.

$$f(I_1, I_2) = \begin{cases} \text{true} & \text{if } I_1 \text{ and } I_2 \text{ are similar} \\ \text{false} & \text{otherwise} \end{cases}$$

Then, similarity retrieval could be performed by retrieving all T_i such that $f(Q, T_i)$ is true.

However, designing a binary similarity function is difficult.

Instead, design a function which computes how similar two images are.

$f(I_1, I_2) = v \in \mathbb{R}$

The more similar I_1 and I_2 are, the larger v.

Two ways to perform QBE retrieval:

1) Using a similarity cutoff.

Specify a similarity cutoff threshold v_t and return all T_i such that $f(Q, T_i) > v_t$.

Ad: Simple - just choose threshold.
Disad: Picking v_t might not be intuitive.
How to determine optimal v_t?

2) By returning the top-N most similar images.

Given N (say 20), rank the target images in order of decreasing similarity using $f(Q, T_i)$ and pick the top N.

Ad: More intuitive to pick N than specify similarity cutoff threshold.
Disad: Not all N might be similar or more than N might be similar.

Note: Sometimes compute "distance" instead of similarity.

The smaller the distance between two images, the more similar.

Eg. Euclidean distance between two feature vectors.

QBE now: 1) Retrieve all target images with a distance less than v_t of Q.

2) Rank target images in order of increasing distance and retrieve top-N.

Distance has a more intuitive interpretation when we represent our images using feature vectors.

within t_p — $t_p - N$

Focus on design of similarity/distance measure (function):

$$d(I1, I2) = v \in \mathbb{R}$$

consider pixel-wise difference.

If I1 and I2 are both M×N pixels:

$$d_{L2}(I1, I2) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} (I1_{m,n} - I2_{m,n})^2$$

similarity or distance measure?

(Squared) Euclidean or L2 distance

$$d_{L1}(I1, I2) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |I1_{m,n} - I2_{m,n}|$$

L1 distance

Adv: Simple and intuitive

Disadv: Not invariant to translation. Single pixel shift can cause large change in output. ("Too local")
(Also not invariant to rotation, scale, intensity scaling.)
Requires that images be of the same size.
Computationally expensive if images are large.

Instead of pixel-wise comparison, compute and compare a representation of the images.

Representation / Feature / Descriptor:

- Typically lower dimension than image
- Captures visual characteristics of image
- Possibly provides invariance to
  - image size
  - translation
  - rotation
  - scale
  - intensity shifts

Visual characteristics of images:

- Color or grayscale values.
- Texture
- Shape - requires boundaries though
- Spatial layout (of regions for example)
- Learned features

Simple descriptor for grayscale images: mean and standard deviation of pixel values.

$$f = (f_0, f_1)$$

image is size $M \times N$

$$f_0 = \mu = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I_{m,n}$$

$$f_1 = \sigma = \left(\frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} (I_{m-n} - \mu)^2 \right)^{1/2}$$

Adv: Simple to compute and low dimensional (2D)
Invariant to translation, rotation

Disad: Does not capture spatial layout of pixel values.

How to compare these features.

Suppose I1 has feature f1
I2 " " f2

Use L2 distance

$$d(I1, I2) = \sqrt{(f1_0 - f2_0)^2 + (f1_1 - f2_1)^2} = \sqrt{\sum_{i=0}^{1} (f1_i - f2_i)^2}$$