# Compression and Indexing of Images for Databases

Gio Wiederhold
CS545I

## Compression ⇔ Indexing .

Image represented by fewer bits
- Less space needed
- Lower transmission load
- Extensive automated pre-processing
- Typically fast restoration

Contents represented by characters
- Much less space needed
- Much easier to transmit
- Difficult semi-automated pre-processing
- No restoration, reference only to source

Both techniques are common, rarely integrated

## Compression methods

- Pixel-sequence
  - sequences of 1’s, 0’s, , , to count
    - *(easy for TV-lines, but sequence lengths are limited)*
  - pattern creation and indexing (Ziff-Lempel)
    - *(less effective without semantic-based repeats, as in text)*
- 2D, 3D Localities
  - Preprocessing to frequency domain
  - Blocks 8 x 8 pixels (JPEG) for images
    - *(some storage and latency in encoding)*
  - Cubes 8 x 8 x 8 (MPEG) for video
    - *(much storage and latency in encoding)*

## Refinements

- Lossy compression: ignore small differences
- Interpolation:
  - transmit complete frame image rarely
    - (MPEG: 3 / second)
  - create morphed frames in the interval
  - (MPEG: midway)
  - correct morphed frame blocks that differ greatly
    - (not now in MPEG)
    - create more morphed frames in interval
    - (MPEG: recursively)

Effect on TV images:

reduction to 5%, imperceptible quality difference

## Transmission

- MPEG developments --> NTSC TV at 650Kb/s
  - extensive preprocessing in transmitter
  - latency in reconstruction in set-top box
- HDTV cannot be compressed that much (1.5Mb/s?)
  - digital format easier to manage
- Video conferencing at 128Kb/s
- Irregular rate due to hard-to-compress images
  - requires additional buffering (10 images?)

## Future Directions

- Segmented compression
  - recognize irregular, similar segments
    - (2D, 3D Ziff-Lempel-like?)
    - arrange in prefix-tree for recovery.
- Scene-based segmentation
  - distinguish background, moving objects
    - background (scanned by viewer), transmit motion, borders
    - objects handled with more fidelity
- Model-based compression
  - create model from image? hard research problem
  - alternate viewers, create *flat* image only at receiver

## Indexing Outline

1. Create sets of symbolic descriptors for images
   - attach references to stored image objects
2. Collect descriptors for easy processing
   - sort, arrange in a hierarchy, intersect, union
3. To retrieve, create descriptor set for query
4. Match descriptors, typically multiple (partial match)
5. Provide feedback to customer
   - too many / displayable sample / exact match / no match
6. Improve the query if needed, return to 3.

## Index features

- Traditional databases:
  - primary index with a unique mapping (good to have)
  - secondary indexes (for non-unique attributes)
    - refer to stored objects (more maintenance as DB is updated)
    - refer to primary index (slower access)
- Bibliographic Databases
  - one super-index (*it's all words*)
  - efficient operations to create unions and intersects
    - needed for synonyms (*fast, speedy*)
    - reduce result, and define domain (*nail, hammer*)
- Image databases need both !

# Index Creation Alternatives

- Index entire image or video clip  (vs segments ➔ ➔ )
  - external descriptors (vs internal features ➔ )
    - obtained when image was acquired
      - name, track, lat, long, viewing angles for  satellite images
      - source, location, time (often irregular data)
    - provided by image analysts
    - captions for images from books, video clips
    - speech recognition (CMU)

# Features for describing images

- internal features
  - formal parameters:
    - Avg.luminosity, granularity, color spectrum,
    - Eigenvalues (used for fingerprints)
  - template match
    - blocks for fields, lines without sharp bends for RR tracks
    - presence of airports
    - presence of airplanes (generalized cones)
  - image understanding
- –
- •

## Segmenting prior to Indexing

- Index segments *(to be more coherent)*
  - segments are created by human outlining
    - with subsequent adjustment
  - segments are human selected from preprocessed images (detail eliminated, candidate bondaries exist)
  - segments are human selected and then processed to locate borders (snakes)
  - segments are due to gridding
    - grid is regular *(hard to match to query)*
    - grid is recursive, terminal when pixels are identical
  - segments are image-based (image processing)
  - segments are model-based (image understanding)

## Features of segments

coherence permits finer descriptions

- number (per type)
- size (or size distribution)
- shape (moments [QBIC])
- orientation
- color (or spectrum)
- texture
- 

each feature creates one (or more) indexes

## Storing Segments

- by reference to the image only
- breaking up the image and placing segments
  - into a grid (blocks arranged by recursive space filling)
    - useful for subsequent image traversal
  - into a tree (but maintain image locality)
    - useful for subsequent image traversal
    - good aid to compression
  - into segment-type specific files
    - fast presentation of objects to consumer (out of context)
    - efficient storage due to homegeneity
    - costly to reconstruct the total images (but only a few should be presented anyhow)

## Index construction

Definitions

- One index lists one attribute type

  Unique index has one distinct key value per image (*n* entries)

  Most image indexes are non-unique (*c* distinct values <<*n* )
- Indexes are sorted to bring like index-entries together
- Indexes can be compressed (prefixes are locally similar)
- Indexes can be
  1. distinct from the data (even remote) (limits compression)
  2. adjoining the data
  3. intermingled (rapid data access)
- Image files will typically have many indexes (*m* )

## Attributes for images/segments

- Darkness (overall) (limited to segments)
- dark/grey/light scale histogram
- Color triple
- Color histogram
- granularity / histogram
- dominant line orientation / histogram
- dominant shape type / histogram
- dominant object type / histogram
- exceptional types (relative to an images series)
  - lightflash (nuclear monitoring)
  - big vehicle
  - many people

## Index usage

- Non-unique indexes must be used in combination to reduce result to a reasonable (if not unique) subset
- Prediction of selectivity for index *1, 2, ..., m* used

  $$\boxed{s = 1/c_1 * 1/c_2 * \ldots * 1/c_m \sim n}$$

  seems easy, but values are very correlated, so that either $s << n$ (too many results) or $s > n$ (none).
- So use some indexes for selection (assuring $s < n$) and others for ranking (close match of index values)

## Index selection

Which indexes to use for

1. match selections *reliable, external*

   take intersections (inputs can be large)

   **(poorly supported by relational, business-oriented systems)**

2. ranking *fuzzy, internal*

   show at least thumbnails, user cannot judge from index values

3. not at all *overly restrictive*

Combine indexes into a hierarchy? (K,D trees)

say, *shape* before *color*

situation, customer dependent

Multiple hierarchies ?

## Cost of dealing with indexes

- more indexes ⇨ linearly more cost
- cost per index varies theoretically, more practically
  - sorted into blocks, sub indexes to blocks, sub-sub accesses = x, $x = \log_y n$ can be few:

    $x$ = no of levels; $y$ = fanout = blocksize/entries (compress!)

    $n$ = 106, $B$= 1000, $E$=10, $y$ = 100 ⇨ 3 ! fast

    - This is true in well-built systems, as most relational, but those do not handle intersection (fetch large subsets)
    - indexes are less well built in bibliographic systems. and those handle intersections well
- Need better infrastructure support

## Fuzzy queries

nearest neighbor semantically

- drop low-order bits if descriptor is continuous
  - complexity: *color represented by a triple*
- provide auxliary mappings for discrete terms

nearness is a semantic concept,
    depends on customer, application

Example `*near SFO*'

- for pilot: {San Carlos, Alameda NAS, Ames, Oakland}
- for airline pilot: {Oakland, San Jose}
- for taxi {Millbrae, Burlingame, South City, San Mateo, Belmont, San Francisco, San Carlos}

## Summary

indexing method effectiveness depends on

- image type (source)
- volume (determines feasibiluty of :
  - manual preprocessing
  - automatic preprocessing with human supervision
  - automatic processing with human exception handling
  - full automatic processing of storage and indexing, human only involved in filtering querry results
- application
  - select nice picture for publication
  - warn people of disasters
  - ....

## Research

- Scalability of nice methods (including those below)
- combinatorial indexing infrastructure
- robust descriptors
- priorities in fuzzy retrieval
- segmentation
- balancing locality withindexing partitioning
- combining compression and indexing effectively
- model-based compression and indexing
- . . .