

## Chapter 18

### Content-Based Retrieval in Digital Libraries

#### [18.1 How Should We Retrieve Images?](#)

#### [18.2 C-BIRD — A Case Study](#)

#### [18.3 Synopsis of Current Image Search Systems](#)

#### [18.4 Relevance Feedback](#)

#### [18.5 Quantifying Results](#)

#### [18.6 Querying on Videos](#)

#### [18.7 Querying on Other Formats](#)

#### [18.8 Outlook for Content-Based Retrieval](#)

#### [18.9 Further Exploration](#)

## 18.1 How Should We Retrieve Images?

- **Text-based search** will do the best job, provided the multimedia database is fully indexed with proper keywords.
- Most multimedia retrieval schemes, however, have moved toward an approach favoring multimedia content itself (“content-based”).
- Many existing systems retrieve images with the following **image features** and/or their variants:
  - **Color histogram**: 3-dimensional array that counts pixels with specific Red, Green, and Blue values in an image.
  - **Color layout**: a simple sketch of where in a checkerboard grid covering the image to look for blue skies or orange sunsets, say.
  - **Texture**: various texture descriptors, typically based on edges in the image.



Fig. 18.1: How can we best characterize the information content of an image? *Courtesy of Museo del Prado.*

## 18.2 C-BIRD — A Case Study

- **C-BIRD** (*Content-Base Image Retrieval from Digital libraries*): an image database search engine devised by one of the authors of this text.
  - [Link to Java applet version of C-BIRD search engine..](#)
- **C-BIRD GUI**: the online image database can be browsed, or searched using a selection of tools: (Fig. 18.2)
  - Text annotations
  - Color histogram
  - Color layout
  - Texture layout
  - Illumination Invariance
  - Object Model

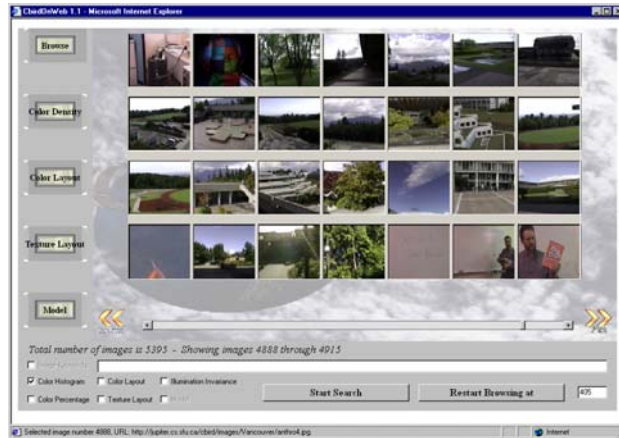


Fig. 18.2: C-BIRD image search GUI.

## Color Histogram

- A color histogram counts pixels with a given pixel value in Red, Green, and Blue (RGB).
- An example of histogram that has  $256^3$  bins, for images with 8-bit values in each of R, G, B:

```
int hist[256][256][256]; // reset to 0
//image is an appropriate struct with byte fields red, green, blue
for i=0..(MAXY-1)
  for j=0..(MAXX-1)
  {
    R = image[i][j].red;
    G = image[i][j].green;
    B = image[i][j].blue;
    hist[R][G][B]++;
  }
```

## Color Histogram (Cont'd)

- Image search is done by matching **feature-vector** (here color histogram) for the sample image with feature-vector for images in the database.
- In C-BIRD, a color histogram is calculated for each target image as a preprocessing step, and then referenced in the database for each user query image.
- For example, Fig. 18.3 shows that the user has selected a particular image — one of a red flower on a green foliage background.

The result obtained, from a database of some 5,000 images, is a set of 60 matching images.



Fig. 18.3: Search by color histogram results.

## Histogram Intersection

- **Histogram intersection:** The standard measure of similarity used for color histograms:
  - A color histogram  $H_i$  is generated for each image  $i$  in the database – feature vector.
  - The histogram is *normalized* so that its sum (now a double) equals unity – effectively removes the *size* of the image.
  - The histogram is then stored in the database.
  - Now suppose we select a *model* image – the new image to match against all possible targets in the database.
  - Its histogram  $H_m$  is intersected with all database image histograms  $H_i$  according to the equation

$$\text{intersection} = \sum_{j=1}^n \min(H_i^j, H_m^j) \quad (18.1)$$

$j$  – histogram bin,  $n$  – total number of bins for each histogram

- The closer the intersection value is to 1, the better the images match.

## Color Density

- The scheme used for showing Color Density is displayed in Fig. 18.4.
- What percentage of the image having any particular color or set of colors is selected by the user, using a color-picker and sliders.
- User can choose from either conjunction (ANDing) or disjunction (ORing) a simple color percentage specification.
- This is a very coarse search method.

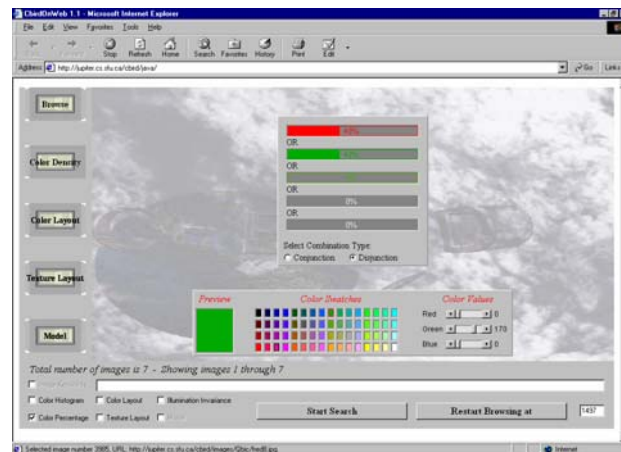


Fig. 18.4: Color density query scheme.

## Color Layout

- The user can set up a scheme of how colors should appear in the image, in terms of coarse blocks of color.
 

The user has a choice of four grid sizes:  $1 \times 1$ ,  $2 \times 2$ ,  $4 \times 4$  and  $8 \times 8$ .
- Search is specified on one of the grid sizes, and the grid can be filled with any RGB color value or no color value at all to indicate the cell should not be considered.
- Every database image is partitioned into windows four times, once for every window size.
  - A clustered color histogram is used inside each window and the five most frequent colors are stored in the database
  - Position and size for each query cell correspond to the position and size of a window in the image
- Fig. 18.5 shows how this layout scheme is used.

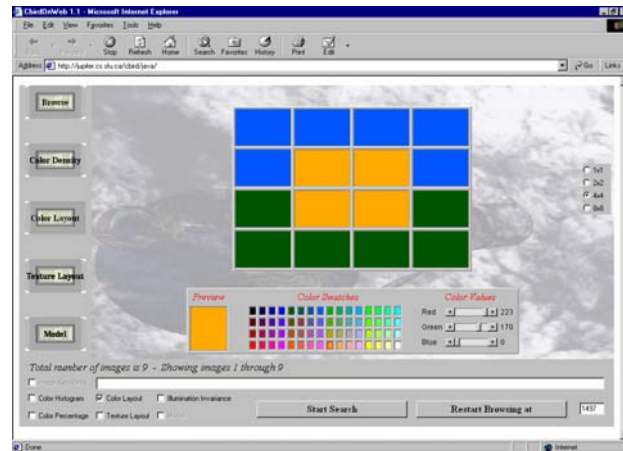


Fig. 18.5: Color layout grid.

## Texture Layout

- This query allows the user to draw the desired texture distribution.
- **Available textures:** zero edge density, medium or high density edges in four directions ( $0^\circ$ ,  $45^\circ$ ,  $90^\circ$ ,  $135^\circ$ ) and combinations of them.
- **Texture matching** is done by classifying textures according to directionality and density (or separation), and evaluating their correspondence to the texture distribution selected by the user.
- Fig. 18.6 shows how this layout scheme is used.

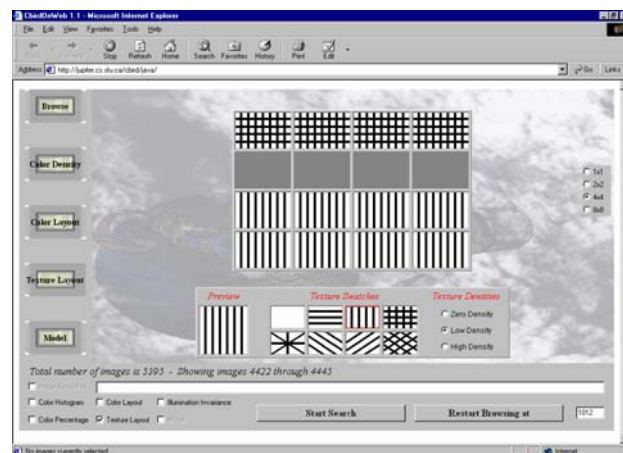


Fig. 18.6: Texture layout grid.

## Texture Analysis Details

### 1. Edge-based texture histogram

- A 2-dimensional texture histogram is used based on edge *directionality*  $\phi$ , and *separation*  $\xi$  (closely related to *repetitiveness*).
- To extract an edge-map for the image, the image is first converted to luminance  $Y$  via  $Y = 0.299R + 0.587G + 0.114B$ .
- A *Sobel edge operator* is applied to the  $Y$ -image by sliding the following  $3 \times 3$  weighting matrices (*convolution masks*) over the image.

$$d_x : \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad d_y : \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (18.2)$$

- The edge magnitude  $D$  and the edge gradient  $\phi$  are given by

$$D = \sqrt{d_x^2 + d_y^2}, \quad \phi = \arctan \frac{d_y}{d_x} \quad (18.3)$$

## Texture Analysis Details (Cont'd)

### 2. Preparation for creation of texture histogram

- The edges are thinned by suppressing all but maximum values.

If a pixel  $i$  with edge gradient  $\phi_i$  and edge magnitude  $D_i$  has a neighbor pixel  $j$  along the direction of  $\phi_i$  with gradient  $\phi_j \approx \phi_i$  and edge magnitude  $D_j > D_i$  then pixel  $i$  is suppressed to 0.

- To make a binary edge image, set all pixels with  $D$  greater than a threshold value to 1 and all others to 0.
- For edge separation  $\xi$ , for each edge pixel  $i$  we measure the distance along its gradient  $\phi_i$  to the nearest pixel  $j$  having  $\phi_j \approx \phi_i$  within  $15^\circ$ .
- If such a pixel  $j$  doesn't exist, then the separation is considered infinite.

## Texture Analysis Details (Cont'd)

### 3. Having created edge directionality and edge separation maps, a 2D texture histogram of $\xi$ versus $\phi$ is constructed.

- The initial histogram size is  $193 \times 180$ , where separation value  $\xi = 193$  is reserved for a separation of infinity (as well as any  $\xi > 192$ ).
- The histogram is "smoothed" by replacing each pixel with a weighted sum of its neighbors, and then reduced to size  $7 \times 8$ , separation value 7 reserved for infinity.
- Finally, the texture histogram is normalized by dividing by the number of pixels in the image segment. It will then be used for matching.

## Search by Illumination Invariance

- To deal with illumination change from the query image to different database images, each color channel band of each image is first normalized, and then compressed to a 36-vector.
- A 2-dimensional color histogram is then created by using the *chromaticity*, which is the set of band ratios

$$\{R, G\} / (R + G + B)$$

- To further reduce the number of vector components, the DCT coefficients for the smaller histogram are calculated and placed in zigzag order, and then all but 36 components dropped.
- Matching is performed in the compressed domain by taking the Euclidean distance between two DCT-compressed 36-component feature vectors.
- Fig. 18.7 shows the results of such a search.



Fig. 18.7: Search with illumination invariance.

## Search by Object Model

- This search type proceeds by the user selecting a thumbnail and clicking the *Model* tab to enter object selection mode.
  - An image region can be selected by using primitive shapes such as a rectangle or an ellipse, a magic wand tool that is basically a seed-based flooding algorithm, an active contour (a “snake”), or a brush tool where the painted region is selected.
  - An object is then interactively selected as a portion of the image.
  - Multiple regions can be dragged to the selection pane, but only the active object in the selection pane will be searched on.
- A sample object selection is shown in Fig. 18.8.

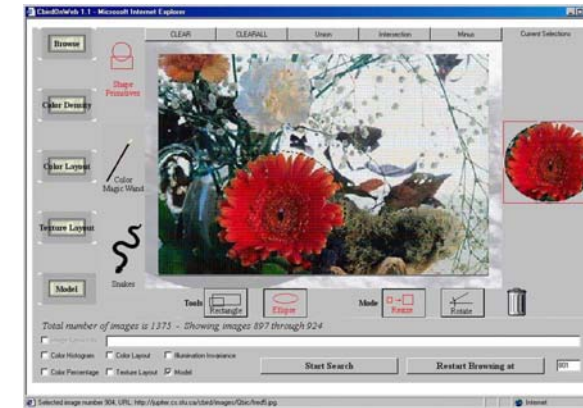


Fig. 18.8: C-BIRD interface showing object selection using an ellipse primitive.

## Details of Search by Object Model

1. The user-selected model image is processed and its features localized (i.e., generate color *locales* [see below]).
2. Color histogram intersection, based on the reduced chromaticity histogram, is then applied as a first *screen*.
3. Estimate the pose (scale, translation, rotation) of the object inside a target image from the database.
4. Verification by intersection of texture histograms, and then a final check using an efficient version of a Generalized Hough Transform for shape verification.

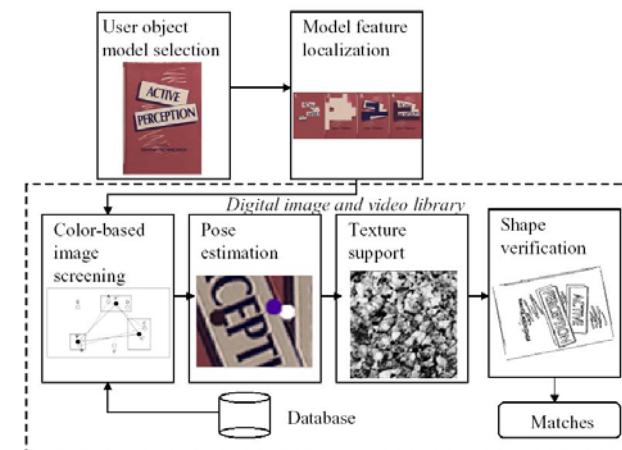


Fig. 18.9: Block diagram of object matching steps.

## Model Image and Target Images

- A possible model image and one of the target images in the database might be as in Fig. 18.10.



Fig. 18.10: Model and target images. (a): Sample model image. (b): Sample database image containing the model book.

## Image Segmentation vs. Feature Localization

- **Image Segmentation:** If  $R$  is a segmented region,
  1.  $R$  is usually connected; all pixels in  $R$  are *connected* (8-connected or 4-connected).
  2.  $R_i \cap R_j = \phi$ ,  $i \neq j$ ; regions are *disjoint*.
  3.  $\cup_{i=1}^n R_i = I$ , where  $I$  is the entire image; the segmentation is *complete*.
- **Feature Localization:** a coarse localization of image features based on proximity and compactness – more effective than Image Segmentation.

## 1. Locales in Feature Localization

- **Definition:** *Locale*  $\mathcal{L}_f$  is a local enclosure of feature  $f$ .
- A locale  $\mathcal{L}_f$  uses blocks of pixels called *tiles* as its positioning units, and has the following descriptors:
  1. Envelope  $L_f$  — a set of tiles representing locality of  $\mathcal{L}_f$ .
  2. Geometric parameters — mass  $M(\mathcal{L}_f) = \text{count of the pixels having feature } f$ , centroid  $\mathbf{C}(\mathcal{L}_f) = \sum_{i=1}^{M(\mathcal{L}_f)} \mathbf{P}_i / M(\mathcal{L}_f)$ , and eccentricity  $E(\mathcal{L}_f) = \sum_{i=1}^{M(\mathcal{L}_f)} \|\mathbf{P}_i - \mathbf{C}(\mathcal{L}_f)\|^2 / M(\mathcal{L}_f)$ .
  3. Color, texture, and shape parameters of the locale. For example, locale chromaticity, elongation, and locale texture histogram.

## Properties of Locales

- After a feature localization process the following can be true:
  1.  $\exists f : \mathcal{L}_f$  is *not connected*.
  2.  $\exists f \exists g : \mathcal{L}_f \cap \mathcal{L}_g \neq \phi$ ,  $f \neq g$ ; locales are *non-disjoint*.
  3.  $\cup_f \mathcal{L}_f \neq I$ , *non-completeness*; not all image pixels are represented.
- Fig. 18.11 shows a sketch of two locales for color red, and one locale for color blue
  - The links represent an association with an *envelope*. Locales do not have to be connected, disjoint or complete, yet colors are still localized.

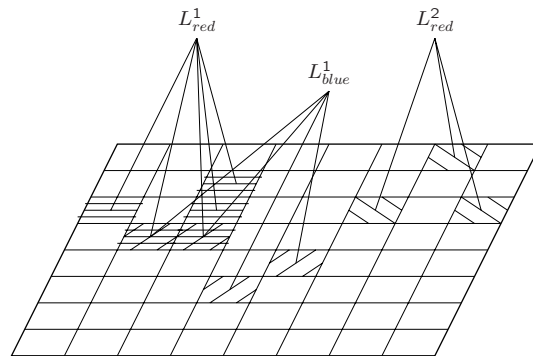


Fig. 18.11: Locales for Feature Localization.

## 2. Tile Classification

- Before locales can be generated, tiles are first classified as having certain features, e.g., red tiles, or red and blue tiles.
- **Dominant color enhancement**
  - Prior to the classification of feature tiles, image pixels are classified as having either *dominant color* or *transitional color*.
  - **Dominant colors:** pixel colors that do not lie on a slope of color change in their pixel neighborhood. *Transitional colors* do.
  - Enhancing the uniformity of the dominant colors is accomplished by smoothing the dominant pixels only, using a  $5 \times 5$  averaging filter, with the proviso that only dominant pixels that have similar color are averaged.

## Dominant Color Enhancement (Cont'd)

- Fig.18.12 shows how dominant color enhancement can clarify the target image in Fig. 18.10.



Fig. 18.12: Smoothing using dominant colors. (a): Original image not smoothed. (b): Smoothed image with transitional colors shown in light gray. (c): Smoothed image with transitional colors shown in the replacement dominant colors (if possible). Lower row shows detail images.

## Tile Feature List

- Tiles have a *tile feature list* of all the color features associated with a tile and their geometrical statistics.
  - On the first pass, dominant pixels are added to the tile feature list
  - On the second pass, all transitional colors are added to the dominant feature list without modifying the color, yet updating the geometrical statistics
  - When all pixels have been added to the tiles, the dominant and transitional color feature lists are merged.



### 3. Locale Generation

- **Locales** are generated using a dynamic  $4 \times 4$  overlapped pyramid linking procedure.

(a). The initialization proceeds as:

```

PROCEDURE 18.1 LocalesInit // Pseudo-code
begin
  Let  $c[n_x][n_y]$  be the 2D array of child nodes.
  Let  $p[n_x/2][n_y/2]$  be the 2D array of parent nodes.
  For each child node  $c[i][j]$  do
    Let  $cn = c[i][j]$  and  $pn = p[i/2][j/2]$ .
    For each node  $cn_p$  in the feature list of  $cn$  do
      Find node  $pn_q$  in the feature list of  $pn$ 
      that has similar color.
      If the merged eccentricity of  $cn_p$  and  $pn_q$  has
       $E < \tau$  then
        Merge  $cn_p$  and  $pn_q$ .
      If  $pn_q$  doesn't exist or  $E \geq \tau$  then
        Add  $cn_p$  to the start of the feature list of  $pn$ .
  end

```

- (b). After the pyramid linkage initialization, parents compete for linkage with each child:

```

PROCEDURE 18.2 EnvelopeGrowing // Pseudo-code
begin
  Let  $c[n_x][n_y]$  be the 2D array of child nodes.
  Let  $p[n_x/2][n_y/2]$  be the 2D array of parent nodes.
  Repeat until parent-child linkage does not change anymore
    For each child node  $c[i][j]$  do
      Let  $cn = c[i][j]$  and  $pn \in p[\frac{i+1}{2}][\frac{j+1}{2}]$ 
      For each node  $cn_p$  in the feature list of  $cn$  do
        Find node  $pn_q$  in the feature lists of  $pn$ 
        that has similar color and minimizes the distance
         $\|C(cn_p) - C(pn_q)\|$ 
        If the merged eccentricity of  $cn_p$  and  $pn_q$  has
         $E < \tau$  then
          Swap the linkage of  $cn_p$  to its parent to  $pn_q$ .
          Update the associated geometrical statistics.
  In the parent feature list  $p$  remove empty nodes.
  Go up a level in the pyramid and repeat the procedure
end

```

- (c). Following the pyramidal linking, locales having small mass are removed and also sorted according to decreasing mass size for efficiency of search

- The color update equation for parent locale  $j$  and child locale  $i$  at iteration  $k + 1$  is

$$\left(r_j^{(k+1)}, g_j^{(k+1)}, I_j^{(k+1)}\right)^T = \frac{\left(r_j^{(k)}, g_j^{(k)}, I_j^{(k)}\right)^T M_j^{(k)} + \left(r_i^{(k)}, g_i^{(k)}, I_i^{(k)}\right)^T M_i^{(k)}}{M_j^{(k)} + M_i^{(k)}} \quad (18.6)$$

- the update equations for the geometrical statistics are

$$M_j^{(k+1)} = M_j^{(k)} + M_i^{(k)} \quad (18.7)$$

$$C_j^{(k+1)} = \frac{C_j^{(k)} M_j^{(k)} + C_i^{(k)} M_i^{(k)}}{M_j^{(k+1)}} \quad (18.8)$$

$$E_j^{(k+1)} = \frac{(E_j^{(k)} + C_{x,j}^{(k)2} + C_{y,j}^{(k)2})M_j^{(k)} + (E_i^{(k)} + C_{x,i}^{(k)2} + C_{y,i}^{(k)2})M_i^{(k)}}{M_j^{(k+1)}} - C_{x,j}^{(k+1)2} - C_{y,j}^{(k+1)2} \quad (18.9)$$



(a)



(b)

Fig. 18.13: Color locales: (a) For the model image. (b) For a database image.

## 4. Texture Analysis

- Every *locale* is associated with a locale-based texture histogram.
  - A *locale-dependent* threshold is better for generating the edge map.
  - The threshold is obtained by examining the histogram of the locale edge magnitudes.
- Global versus locale-based texture measures:
 

The locale-based texture is a more effective measure of texture than is a global one, since the locale-dependent thresholds can be adjusted adaptively.

  - Fig. 18.14 compares a locale-based edge-detection to a global threshold based edge-detection.

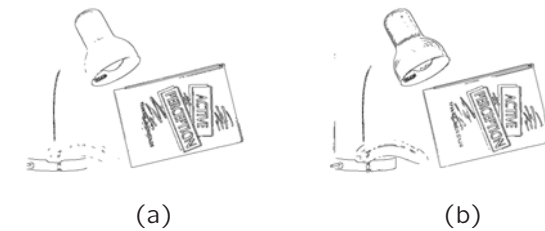


Fig. 18.14: Global versus locale-based thresholds. (a) The edge-map for the database image using a global threshold. (b) The edge-map for the database image using a locale-based threshold.

## 5. Object Modeling and Matching

- The object image selected by the user is sent to the server for matching against the locales database.
- **Locale Assignment:** the one-to-one correspondence between image locales to be found and model locales.
  - A locale assignment has to pass several screening tests to verify an object match.
  - Screening tests are applied in order of increasing complexity and dependence on previous tests.
- The sequence of steps during an object matching process is shown in Fig. 18.9.
  - (a) user object model selection and model feature localization
  - (b) color-based screening test
  - (c) pose estimation
  - (d) texture support
  - (e) shape verification

## Object Match Measure $Q$

- The object match measure  $Q$  is formulated as follows:

$$Q = n \sum_{i=1}^m w_i Q_i \quad (18.10)$$

$n$  – the number of locales in the assignment  
 $m$  – the number of screening tests considered for the measure  
 $Q_i$  – the fitness value of the assignment in screening test  $i$   
 $w_i$  – weights that correspond to the importance of the fitness value of each screening test

- Locales with higher mass (more pixels) statistically have smaller percentage of localization error.
- Assignments with many model locales are preferable to few model locales, since the cumulative locale mass is larger and the errors average out.
- One tries to assign as many locales as possible first, then compute the match measure and check the error using a tight threshold.

## Matching Steps

- The screening tests that are applied to locales in order to generate assignments and validate them are:
  - Color-based screening tests (Step b):
    - \* Illumination Color Covariant Screening
    - \* Chromaticity Voting
    - \* Elastic Correlation
  - Estimation of Image Object Pose (Step c)
  - Texture Support (Step d)
  - Shape Verification (Step e)
  - Recovery of Lighting Change

## Elastic Correlation

- **Elastic Correlation:** the operation that computes the probability that there can be a correct assignment, and returns the set of possible assignments.
  - Can be used to evaluate the feasibility of having an assignment of image locales to model locales by using chromaticity shift parameters.
  - Having a candidate set of chromaticity shift parameters, each candidate is successively utilized for computing the elastic correlation measure.
  - If the measure is high enough (higher than 80%, say), then the possible assignments returned by the elastic correlation process are tested for object matching using pose estimation, texture support and shape verification.

## Elastic Correlation (Cont'd)

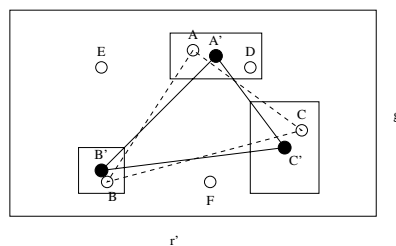


Fig. 18.15: Elastic correlation in  $\Omega\{r', g'\}$

- Fig. 18.15 shows the elastic correlation process applied in the model chromaticity space  $\Omega\{r', g'\}$ :
  - The model image has three locale colors located at  $A'$ ,  $B'$  and  $C'$ .
  - All the image locale colors,  $A$ ,  $B$ ,  $C$ ,  $D$ ,  $E$  and  $F$ , are shifted to the model illuminant.
  - Elastic correlation is employed, in which the nodes  $A$ ,  $B$ ,  $C$  are allowed to be located in the vicinity of  $A'$ ,  $B'$ ,  $C'$ , respectively.

## Pose Estimation Method & Texture Support

- The pose estimation method (Step (c)) uses geometrical relationships between locales for establishing pose parameters.
  - Performed on a feasible locale assignment.
  - Locale spatial relationships are represented by relationships between their centroids.
  - Results of pose estimation are both the best pose parameters for an assignment and the minimization objective value.
  - If the error is within a small threshold, then the pose estimate is accepted.
- The texture support screening test utilizes a variation of histogram intersection technique, intersecting texture histograms of locales in the assignment.
  - If the intersection measure is higher than a threshold then the texture match is accepted.

## Shape Verification

- **Shape verification** is by the method of Generalized Hough Transform (GHT):
  - After performing pose estimation, GHT search reduces to a mere confirmation that the number of votes in a small neighborhood around the reference point is indicative of a match.
  - The reference point used is the model center since it minimizes voting error caused by errors in edge gradient measurements.
- Once we have shape verification, the image is reported as a match, and its match measure  $Q$  returned, if  $Q$  is large enough.
  - Fig.18.16(a) shows the GHT voting result for searching for a model object (pink book, here) from one of the database images as in Fig. 18.16(b).
  - Fig.18.16(b) shows the reconstructed edge map for the book.

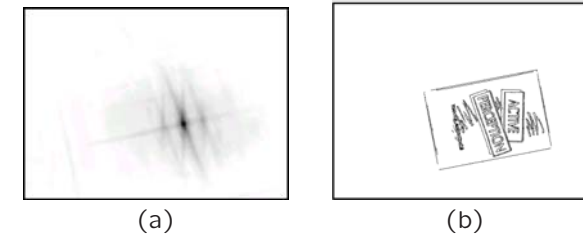
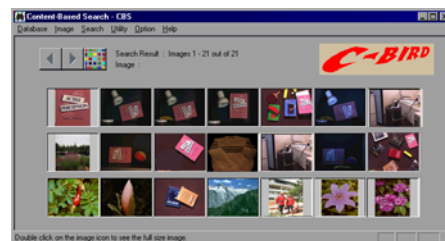
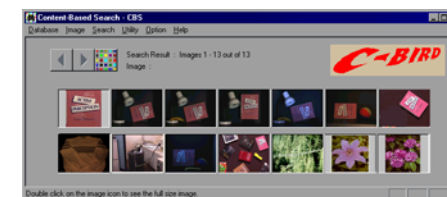


Fig. 18.16: Using the GHT for Shape Verification. (a): GHT accumulator array image. (b): Reconstruction of the detected object using the estimated pose and GHT template (edge map).

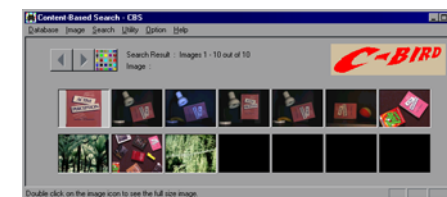


(a)

Fig. 18.17: Search result for the pink book model with illumination change support. (a): Search results using pose estimation only.



(b)



(c)

Fig. 18.17 (cont'd): (b): Search results using pose estimation and texture support. (c): Search results using GHT shape verification.

## 6. Video Locales

- **Video Locales:** a sequence of video frame feature locales that share similar features in the spatio-temporal domain of videos.
  - Like locales in images, video locales have color, texture, and geometric properties.
  - They capture motion parameters such as the motion trajectory and speed, as well as temporal information such as the life-span of the video locale and its temporal relationships with respect to other video locales.
- Fig.18.18 shows that while speeding up the generation of locales substantially, very little difference occurs in generation of locales from each image (*Intra-frame*) and from predicting and then refining the locales (*Inter-frame*).

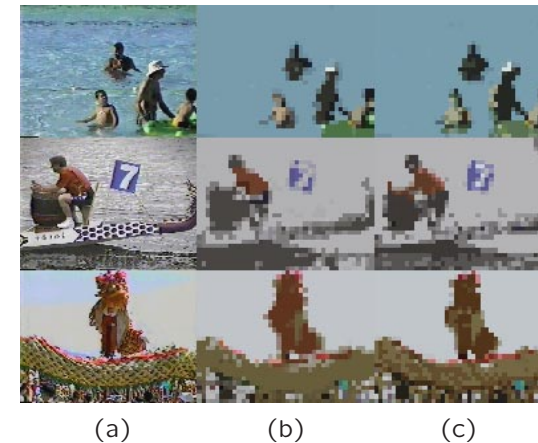


Fig. 18.18: Intra-frame and Inter-frame video locales algorithm results. (a) Original images. (b) Intra-frame results. (c) Inter-frame results.

## 18.3 Synopsis of Current Image Search Systems

- Some well-known current image search engines are listed here. For URLs and resources, please refer to *Further Exploration*.
  - [Link to Further Exploration for Chapter 18.](#)
  - QBIC (Query By Image Content)
  - UC Santa Barbara Search Engines
  - Berkeley Digital Library Project
  - Chabot
  - Blobworld
  - Columbia University Image Seekers
  - Informedia
  - MetaSEEk
  - Photobook and FourEyes
  - MARS
  - Virage
  - Viper
  - Visual RetrievalWare

## 18.4 Relevance Feedback

- **Relevance Feedback:** involve the user in a loop, whereby images retrieved are used in further rounds of convergence onto correct returns.
- **Relevance Feedback Approaches**
  - The usual situation: the user identifies images as good, bad, or don't care, and weighting systems are updated according to this user guidance.
  - Another approach is to move the query towards positively marked content.
  - An even more interesting idea is to move every data point in a disciplined way, by warping the space of feature points.

## Relevance Feedback (cont'd)

- The basic advantage of putting the user into the loop by using relevance feedback is that the user need not provide a completely accurate initial query.
- For a specific example of relevance feedback with respect to the image search engine **Mars**, please refer to the URL in *Further Explorations*.

## 18.5 Quantifying Results

- **Precision** is the percentage of relevant documents retrieved compared to the number of all the documents retrieved.

$$\text{Precision} = \frac{\text{Desired images returned}}{\text{All retrieved images}} \quad (18.13)$$

- **Recall** is the percentage of relevant documents retrieved out of all relevant documents.

$$\text{Recall} = \frac{\text{Desired images returned}}{\text{All desired images}} \quad (18.13)$$

- These measures are affected by the database size and the amount of similar information in the database, and as well they do not consider fuzzy matching or search result ordering.

## 18.6 Querying on Videos

- **Video indexing** can make use of *motion* as the salient feature of temporally changing images for various types of query.
- *Inverse Hollywood*: can we recover the video director's "flowchart"?
  - Dividing the video into *shots*, where each shot consists roughly of the video frames between the on and off clicks of the record button.
  - Detection of shot boundaries is usually not simple as fade-in, fade-out, dissolve, wipe, etc. may often be involved.

- In dealing with digital video, it is desirable to avoid uncompressing MPEG files.
  - A simple approach to this idea is to uncompress only enough to recover just the DC term, thus generating a thumbnail that is 64 times as small as the original.
  - Once DC frames are obtained from the whole video, many different approaches have been used for finding shot boundaries – based on **features** such as color, texture, and motion vectors.

## Video (Temporal) Segmentation

- Shots are grouped into **scenes** — a collection of shots that belong together, and are contiguous in time.
- Even higher-level semantics exists in so-called *film grammar*.
- Audio information is very important for video segmentation.
  - In a typical scene, the audio has no break within a scene, even though many different shots may be taking place over the course of the scene.
- Text may indeed be a useful means of delineating shots and scenes, making use of closed-captioning information already available.
  - Relying on text is unreliable since it may not exist, especially for legacy video.

## Schemes for Organizing and Displaying Storyboards

- The most straightforward method is to display a 2-dimensional array of *keyframes*.
  - Usually a clustering method is used to represent a longer period of time that is more or less the same within the temporal period belonging to a single keyframe.
- Some researchers have suggested using a *graph-based method*.
  - A sensible representation of two talking heads might be a digraph with directed arcs taking us from one person to the other, and then back again.
- Other *proxies* have also been developed for representing shots and scenes.
  - Annotation by text or voice, of each set of keyframes in a **skimmed** video, may be required for sensible understanding of the underlying video.

## An Example of Querying on Video

- Fig. 18.19: (a) shows a selection of frames from a video of beach activity.

Here the keyframes in Fig. 18.19 (b) are selected based mainly on color information (but being careful with respect to the changes incurred by changing illumination conditions when videos are shot).

- A more difficult problem arises when changes between shots are gradual, and when colors are rather similar overall, as in Fig. 18.20(a).

The keyframes in Fig. 18.20(b) are sufficient to show the development of the whole video sequence.



(a)



(b)

Fig. 18.19: Digital video and associated keyframes, *beach* video. (a): Frames from a digital video. (b): Keyframes selected.



(a)



(b)

Fig. 18.20: Garden video. (a): Frames from a digital video. (b): Keyframes selected.

## 18.7 Querying on Other Formats

- A good introduction to using both audio and video cues is the article in text reference [38].
- An interesting effort to understand and navigate slides from lectures, based on the time spent on each slide and the speaker's intonation, is in text reference [39].
- A good introduction to search-by-audio is in text reference [40], and another very interesting approach called "Query-by-Humming" is in text reference [41].
- Other features considered for indexing include indexing on actions, indexing concepts and feelings, indexing facial expressions, and so on. Clearly, this field is a developing and growing one, particularly because of the advent of the MPEG-7 standard (see Chapter 12).

## 18.8 Outlook for Content-Based Retrieval

- The present and future trends identified: indexing, search, query, and retrieval of multimedia data based on:
  1. Video retrieval using video features: image color and object shape, video segmentation, video keyframes, scene analysis, structure of objects, motion vectors, optical flow (from Computer Vision), multispectral data, and so-called 'signatures' that summarize the data.
  2. Use of spatio-temporal queries, such as trajectories.
  3. Semantic features; syntactic descriptors.
  4. Use of relevance feedback, a well-known technique from information retrieval.

5. Retrieval using sound, especially spoken documents, e.g. using speaker information.
6. Multimedia database techniques, such as using relational databases of images.
7. Fusion of textual, visual, and speech cues.
8. Automatic and instant video manipulation; user-enabled editing of multimedia databases.
9. Multimedia security, hiding, and authentication techniques such as watermarking.



## 18.8 Outlook for Content-Based Retrieval

(Cont'd)

- Some other directions:
  - Researchers try to create a search profile so as to encompass most instances available, say all “animals” .
  - **Query-based learning:** intelligent search engines are used to learn a user’s query concepts through active learning for searches using visual features.
  - **Perceptual similarity measure:** focuses on comprehending how images are viewed as similar, by people, on the basis of perception.

## 18.9 Further Exploration

→ [Link to Further Exploration for Chapter 18.](#)

- Links to CBR sites are collected in Chapter 18 Further Exploration section of the text website:
  - A Java applet version of the C-BIRD system utilized in Section 18.2.
  - A demo of QBIC as an artwork server.
  - A demo version of the Alexandria Digital Library.
  - A demo of the Berkeley Digital Library Project.
  - A demo version of VisualSEEK.
  - The Informedia project.
  - A demo of the NETRA system.
  - A demo of Photobook.
  - A demo of the Visual RetrievalWare system.
  - A video describing the technology for the Virage system.
  - A demo site for the VIPER system.
  - Links to standard sets of digital images and videos, for testing of retrieval and video segmentation programs.
  - A link to the keyframe production method for Figs. 18.19, 18.20.