Chapter 2

Background

In this chapter, we describe the fundamental concepts of a content-based image search system and digital rights management mechanisms.

2.1 Content-based Image Search Systems

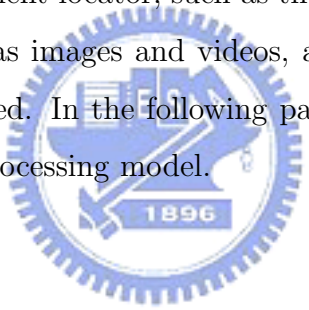
A content-based image search system integrates techniques originally developed in several fields, such as signal processing, image processing, computer vision, pattern recognition, expert system, and more. In this section, we describe the processing model, the concept of relevance feedback, and the performance measures.

2.1.1 Content-based Image Retrieval Model

There are several ways to allocate a specific digital data item. File name, which may be the earliest and the most basic mechanism to identify a piece of data in a computer disk, is a simple and flexible scheme for locating (naming) a file. A simple file search tool in the early days may be designed based on file names. However, the shortcoming of file names is that its capability is rather limited, because most of the operating systems does not allow very long file names. The birth of relational database (RDB) reveals another addressing scheme, indexing by certain predefined

attributes. For example, an academic paper could be indexed by its author names, title, date, and keywords. When implementing in an RDB, these attributes are often stored in different entity fields; when implementing in a local file system, these attributes may be stored in an associated meta file; when implementing in a web system, these attributes may reside in the web page, marked with special HTML tags. In short, this scheme addresses the digital item using *meta-data*. Here the meta-data are keyword-based. Thus we may search for the target item by matching its textual descriptions.

Meta-data could be generated manually or automatically. For a textual document, there are many mature analysis algorithms to extract related keywords from the content. By properly arranging the meta-data [1], a search engine can be a powerful content-based document locator, such as the Google search engine [2]. For non-textual documents such as images and videos, automatic production of meta-data is much more complicated. In the following paragraphs, we give an overview of the content-based image processing model.



Data Flow

A typical image processing system consists of several processing components. They are applied to various aspects of the input images, from signal-level to semantic-level. Fig. 2.1 is one of the generic digital image processing architecture proposed in [3].

For a specific image processing problem, the first step of the solution is *image acquisition*. To obtain a digital image, we first collect data from a set of sensors. The sensors could be a camera, a video recorder, or an infrared telescopes. If the obtained image data are in analog form, a subsequent digitizing process should be followed. In terms of signal processing, this step is an analog-to-digital process.

After we have the digitized images, the next step is to *pre-process* the image data. The key to this function is to improve the obtained data in ways that the following steps are more probable to maximize their performance. Thus, it is not

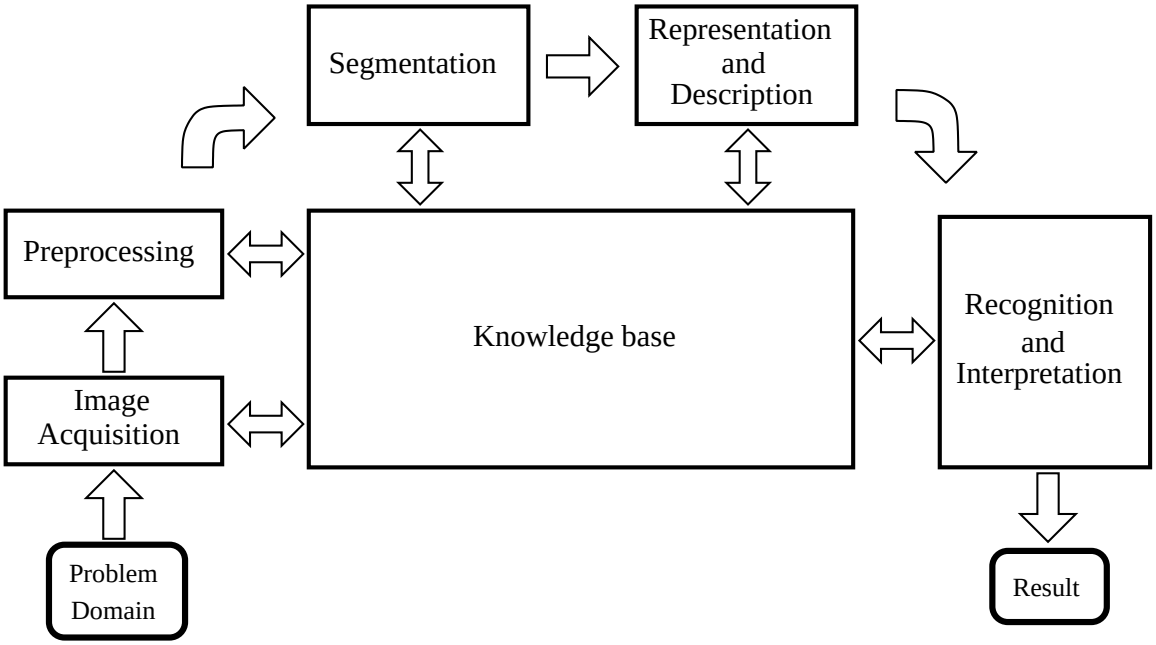


Figure 2.1: Fundamental steps in digital image processing.[3]

surprising that many signal processing techniques could be used in this step. For example, we may reduce the noise and enhance the contrast of an image here to help the extraction of object edges.

The next step deals with *segmentation*. A loose definition of segmentation is to partition an image into its constitution parts. For example, an image of a living room may be segmented as: a wall (background), a paintings on the wall, a chair, a TV set, and etc. In general, to automatically segment an arbitrary image is very difficult. Most successful segmentation methods are applied to a controlled environment in which the models of the partitions (such as object models) are available. This step is critical to the high-level image processing. On the one hand, a good segmentation often leads to a successful solution; on the other hand, a poor or an erroneous segmentation almost guarantees the failures of the high-level analysis of an image.

The segmented results are then sent to the next stage, *representation and description*. The function of this step is to transform the segmented image data into a

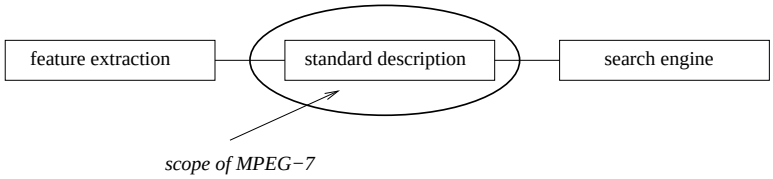


Figure 2.2: The scope of the MPEG-7 standard. [4]

domain which is suitable for efficient computer processing or storage. Although the pre-processing and the segmentation may be also computer processing stages, they incorporate more signal processing oriented algorithms. Starting from this stage, we put more focus on the medium to high semantic level processing. The first task is to transform the data into a different domain if necessary. For example, a segmented area may be represented as a bounding box with opaque and transparent pixels inside, or be represented as a shape with enclosed pixels.

In addition to data representation, another task of transformation is description. Description, in the sense of feature selection, is a method to bring up the specific aspects of the image to our interests. It is often a set of quantitative values, and can be used to differentiate one class of targets from another. To solve image retrieval problems, this is one of the basic tools (and probably the centric one) to determine the similarity of two candidate images. The MPEG-7, also known as *Multimedia Content Description Interface*, is a standard for describing multimedia data. It focuses on the format of the description of content. The associated operations such as feature extractions and search operations are not parts of the standard (Fig. 2.2).

The last stage is *recognition and interpretation*. Recognition is the process that assigns labels to each items based on the extracted descriptions; interpretation is the process that assigns meanings to a set of items. For example, we may first assign the corresponding character (the label) to each recognized letter or digit on an envelope. Then, we detect the spatially close digits in a row, and assign this group as the "zip code" (the interpretation).

So far we have said nothing about the *knowledge base*. As shown in Fig. 2.1, it interacts to every processing modules. We may treat it as a database containing the necessary information for each processing unit. In a very simple system, there may be no such a database because each processing module does not require external information. However, in a more complex system, a knowledge base may provide the modeling information and/or the accumulative information from the previous processing results. It can be even more complex, for example, to include an inference engine to enhance the decision of the recognition and interpretation step, and feedback the results to the knowledge base to improve future segmentation.

Techniques

In the **pre-processing** step, to enhance the results of a query, common techniques used here are image enhancements, geometry transformations, signal space transformations, and others. Color image processing is probably the most active research area in image retrieval. A color image provides more information than its gray-level version, and potentially provides better discrimination, such as object boundaries or depths. Shape analysis is another important tool for pre-processing. It helps us to identify various geometric details in the image. Note that in this step, we try to identify the "local shape" properties, not necessarily the object shape properties. Typical tools used for this purpose are directional filters [5], curvature analysis, edge detectors, and others. In computer vision, texture is defined as the third key element of an image next to color and shape. Because texture properties is very diverse, we cannot assume that they can be represented by a common parameter (feature). The associated research technique is mostly statistical or generative methods. The commonly used tools for texture analysis are Markovian analysis [6], wavelets, and others. To sum up, pre-processing is critical for image retrieval because all the subsequent analysis and interpretation rely on the quality of the pre-processing outputs.

The **image features** (and semantics) are extracted from the pre-processed data, and are arranged in a structural form for further use. The process is mainly a two-

stage operation. The first stage groups data according to their spatial relationships. The *strong segmentation* groups data into regions; each of them is a real-world object. Obviously, the algorithm must be very smart so that the segmented results match the semantics. To overcome the brittleness of strong segmentation, *weak segmentation* may be used to group homogeneous [7] data into conspicuous regions. The weakest tool is the *image partitioning*, which groups data without considering the physical meaning of objects. For example, we can partition an image into the central 90% rectangular area and the outer 10% area. Another case is the *sign detection*. A sign has a well-defined object model and semantics, for example, the traffic signs on a map. When it is applicable, this tool results in the most matching confidence by its unambiguous semantics. After grouping the data in an image, we next try to reduce the amount of data and impose structures on the features. The types of features are:

- the *global and accumulating features* are low-level, such as histograms, which loses the location information of the original image;
- the *salient features* are derived from the weak segmentation results, which represent the salient and robust aspects of an image;
- the *shape and object features* are often the best features for matching, which represents the aspects closest to human semantics;
- The *sign feature* [8] is based on the sign detection, and the location is the key of this kind of features.

Lastly, a practical application often combines various features in a well-organized manner. This produces a more complex description of an image, expressed in the hierarchy or layout of the features.

For content-based image retrieval, the major task is to **determine** how similar two images are. Since we have reduced the content of an image into a structure of features, the comparison is equivalent to **interpreting** features from two images.

The straight-forward case is the unilateral interpretation of sign features, because signs have rigid semantic meanings. However, most CBIR system designers try to resolve real-world ambiguities. In other words, a typical CBIR system often includes similarity-oriented feature matching algorithms. According to the feature type, the similarity metric can be computed in various ways, such as distance-like approach [9], scale space [10] similarity, Bayesian classifiers, or topological approaches. Among all matching schemes, the knowledge-base assisted matching is probably the ideal image searching scheme. If each image in the database has been associated with a set of accurate textual descriptions, the matching can be based on the ontology approaches. Otherwise, this problem becomes much more difficult to solve. To build knowledge from a large collection of images, one approach is to perform cluster analysis. Several researches acquire knowledge from users by *relevance feedback* (will be discussed in Sec. 2.1.2). The relationships among features are determined by the input (feedback) of the user. Another approach is to incorporate learning mechanisms. It is useful when the database is very large and growing. Either supervised or unsupervised learning methods can be included to build the knowledge of the database.

Interacting with users using a data set has been studied in many categorical information retrieval systems. When adopting it into a CBIR system, the problem should be re-considered, because the semantic gap can only be reduced in the query context. Thus, image retrieval requires more *user participation* than the conventional classification problems. By defining an image *query space* Q with four elements: a set of images I_Q, a selected set of features F_Q, a selected set of similarity measures S_Q, and a set of semantics Z_Q, a CBIR system could define its *query specification*. Then, the system can perform *exact query* or *approximate query* according to the specification. The former can be achieved by *spatial predicate*, by *image predicate*, or by *group predicate*. The latter can be achieved by *spatial example* [11], *image example* [12], or *group example*. In addition to the interaction for query, the *query space display* is also an important interaction. Although it does not directly affect the accuracy of a query, a good design in displaying results would

provide intuitive, friendly, and efficient user interface. In some cases, a good display design could reduce the total iterations of user feedbacks [13].

The last category of CBIR techniques is related to the **system aspects**. The first issue is storage and indexing problem. For a large collection of images and their associated features, a structure for maintaining data persistence and retrieval efficiency is very important. However, all high dimensional data suffer from *the curse of dimensionality*. This problem is mainly dealt in three directions: space partitioning, data partitioning, and distance-based techniques. There have been several developed structures for storing high dimensional data. For example the K-D tree can be used with feature space partitioning; the R-tree can be used with data partitioning; and the M-tree can be used with distance-based index structure. The second issue is how to evaluate query accuracy. We will discuss this topic in Sec. 2.1.3.

2.1.2 Relevance Feedback

In this section, we briefly describe the origin of the relevance feedback concept. First, we take a look into how a successful image retrieval can be produced. A typical content-based retrieval can be thought as the result of a series of comparisons. These comparisons may be a mixture of conditions matching of different knowledge domains, as illustrated by Fig. 2.3. The literal equality and similarity define the relation at the signal level regardless of the physical or perceptual meanings. For example, a blue sky and a blue blanket are considered similar in this case. The perceptual comparisons are based on what/how human experiences are applied to the presented signals. The knowledge of colors is a good example. A person may consider two colors similar under a certain condition, while the signal level representation shows that the two colors are quite different. Physical laws describe the sensing and object surface properties. Many general laws of physics may be used to model the target, such as luminance, refection, and color saturation. The geometric

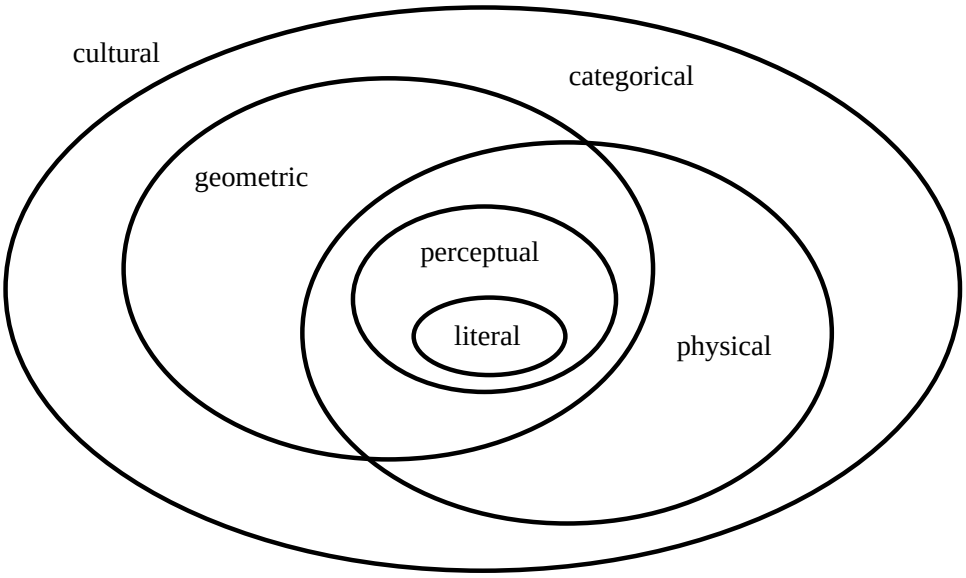


Figure 2.3: Sources of general knowledge ordered as classes of equality.[14]

rules (including the topological properties) describe the relationships among object in the spatial domain. Sometimes it couples with the physical knowledge (e.g., the target object is a part of another object), and sometimes it is independent of the physical knowledge (e.g., only the location of the target object). The categorical rules associate (or tag) the comparison objects with one or more categories. This is a rather high-level process and often is used in a narrowly defined problem domain. For example, a category "teapot" contains teapots of various shape, color, and size. If the criterion is too loose, the categorical analysis is likely to fail. The last set of knowledge, the cultural rules, defines the equality and difference. For example, language-based matching may differ from a geographical region to another in the world.

Since a matching process may deal with several knowledge domains, a CBIR system is required to provide sufficient knowledge when judging "similarity". In terms of image features, features can be roughly categorized into two groups according to their lifetime. The first group is used to describe the static information of an image, such as the color and the textual description; the second group is used to de-

scribe the run-time properties of a content-based processing system, such as the user preferences and the content distribution in a database. Although these two kinds of features represent different aspects of a CBIR problem, they are often coupled together. A CBIR application is often designed for a specific problem domain, and can only process a limit number of static features. Based on these selected features, it incorporates means to conjecture about user preferences for each query. There are no general rules in acquiring user preferences; thus, many CBIR systems have been proposed to interact with users by employing relevance feedback.

A typical Query-by-Example (QBE) CBIR system with relevance feedback generally analyzes the user query images and/or relevant feedback images to derive the necessary search parameters. The search parameters are often defined in terms of the image features pre-chosen in the system. Then, the system searches the database and returns a list of the top-N similar images for further feedback actions. This process can be repeated and hopefully it will eventually produce satisfactory results to that particular user and query. If we treat the relevance feedback function as a means to obtain multiple user inputs, the system can be generalized to an architecture which:

- acquires multiple query instances (positive ones and negative ones);
- analyzes the collected instances based on the pre-chosen features;
- makes a "guess" on the user intention, represented by the pre-chosen features;
- performs the search according to the derived user intention.

The QBIC System

Often, a relevance feedback capable system incorporates algorithms to process multiple input and multiple features. There have been many developed CBIR systems with these functionality to various degrees. QBIC [11] is a well-known work which deals with multiple image/video features. Figure 2.4 is a simplified processing model

of QBIC. The upper part is the analysis process ("database population" in QBIC). It extracts various kinds of features from the given input data, including images, videos, and/or representative frames (r-frames), and store them to a database.

The lower part of Fig. 2.4 is the query process. It includes a query interface which helps users to generate features for the query. By integrating several selection and composition tools, the query interface supports synthetic query input such as query by color histogram, query by texture, and query by sketch. The query features are then sent to the match engine. It computes the distances among features, and lists the best matches in similarity order. Note that in QBIC, it does not try to resolve the semantics of these features. The system only compares the feature vectors in the matching space, and consider a short distance to be similar. The philosophy of the QBIC engine is [11]:

> *... Humans are much better than computers at extracting semantic descriptions from pictures. Computers, however, are better than humans at measuring properties and relating these in long-term memory.*
>
> *One of the guiding principles used by QBIC is to let computers do what they do best–quantifiable measurement– and let humans do what they do best–attaching semantic meaning. ...*

The MARS System

The MARS system [15] is a step toward a relevance feedback capable system. Because it interacts with users to acquire the run-time preferences for each query, it does not provide many tools to present the query interface. The most complex (probably) tool for users is a 5-level scoring selector for each matched image. Users can specify the relevance for an image, feedback to the system, and query again. The system would conduct the feature weightings from the relevant (both positive and negative) images, and hopefully produce better matching results. Relevance feedback is an improvement of query interface in that it eliminates the technical

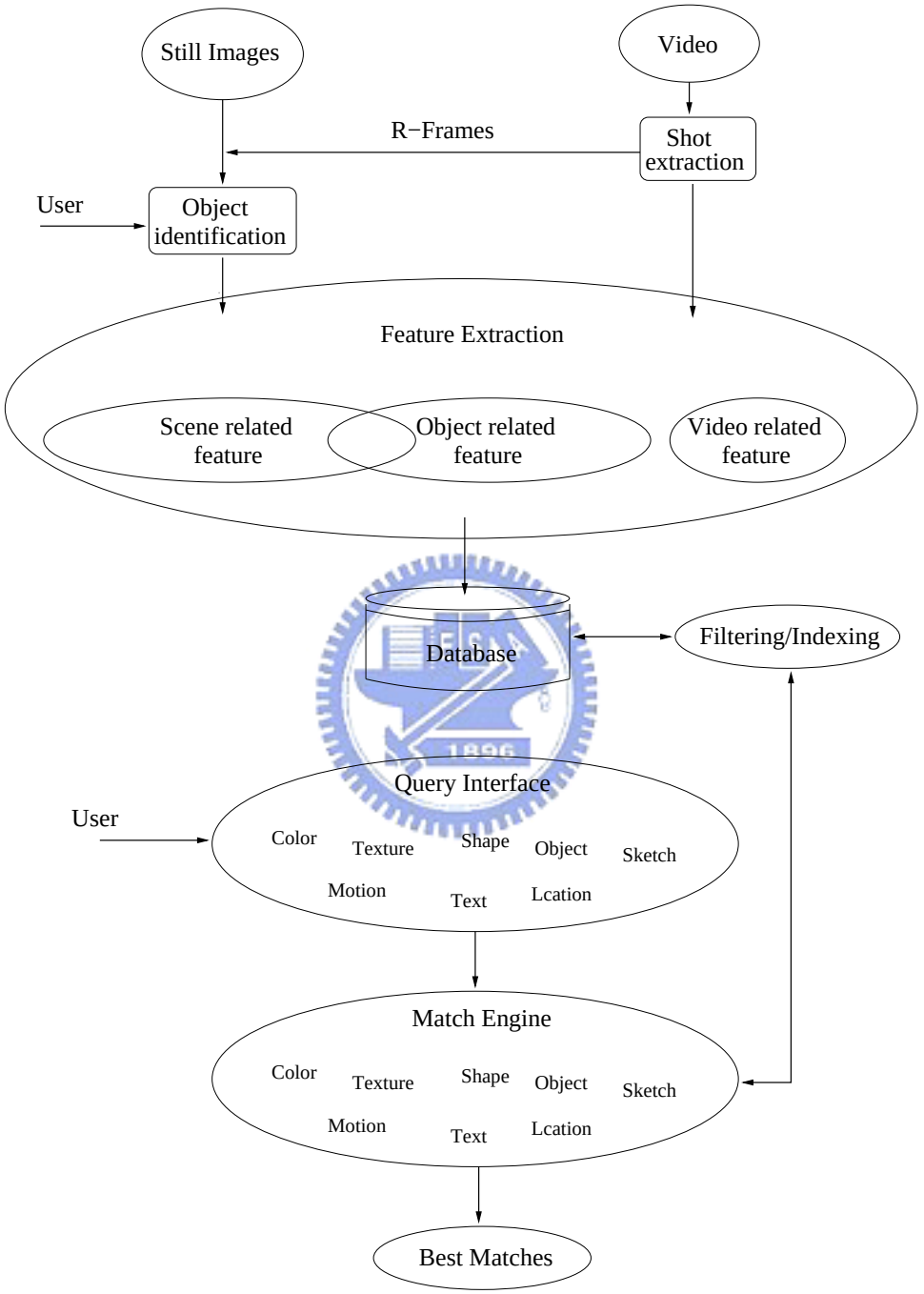


Figure 2.4: A simplified QBIC architecture.

operations. For a non-expert user, some of the query tools are too technical to be used intuitively, such as the sketch tool or the texture tool in QBIC.

Here we briefly describe the matching method used by the MARS. The system organize image features in two levels. The first level is called *feature*. It is a set ($F = \{f_i\}$) of low-level visual features associated with the image, such as color, texture, and shape. The second level is called *representation*. It is a set of representations of a given feature ($R = \{r_{ij}\}$, where r_{ij} is the j-th representation of feature f_i). For example, both color histogram and color moments are representations for the color feature. The goal of relevance feedback is to find the appropriate weighting factors to model the query preferences. There are three kinds of weighting factors: the weighting W_i for each feature f_i, the weighting W_{ij} for each representation r_{ij}, and the weighting W_{ijk} for each component v_{ijk} of a representation r_{ij}.

With a set of similarity measures $M = \{m_{ij}\}$, the similarity function is defined as follows (the object references are omitted). The similarity in terms of r_{ij} is defined by the measure, the representation specification r_{ij}, and all the weights of v_{ijk}:

$$S(r_{ij}) = m_{ij}(r_{ij}, \{W_{ijk} | \forall k\}),$$

and the feature similarity is defined by the weighted sum of all the representation similarity:

$$S(f_i) = \sum_j W_{ij} S(r_{ij}).$$

Finally, the overall similarity is obtained by:

$$S = \sum_i W_i S(f_i).$$

Since the above similarity is a linear combination, we can integrate W_i into W_{ij} and have the form below:

$$S = \sum_i \sum_j W_{ij} S(r_{ij}).$$

To assure the values in all measures having the same dynamic range, we need to perform two kinds of normalization. The first is called *intra-normalization*. It

normalizes each component v_{ijk} of a representation (r_{ij}). The first step is to compute

$$v'_{ijk} = \frac{v_{ijk} - \mu_{ijk}}{\sigma_{ijk}},$$

where μ_{ijk} and σ_{ijk} are the mean and standard deviation over all the image instances in the database. According to the Gaussian distribution property, 68% of the values are in $[-1, 1]$. The second step is to map out-of-range values to either -1 or 1.

The other kind of normalization, called *inter-normalization*, ensures equal emphasis on each $S(r_{ij})$. Before the normalization starts, we compute the mean (μ_{ij}) and the standard deviation (σ_{ij}) over all possible pair of r_{ij}. Then, the similarity is adjusted:

$$S'(r_{ij}) = \frac{S(r_{ij}) - \mu_{ij}}{3\sigma_{ij}}.$$

This guarantees that 99% of $S'(r_{ij})$ are in $[-1, 1]$. An additional shifting ensures that 99% values are in the range $[0, 1]$:

$$S''(r_{ij}) = \frac{S'(r_{ij}) + 1}{2}.$$

Because similarity values should be non-negative, the final step is to map all out-of-range values to either 0 or 1.

After the values are normalized, we then update weights. For *intra-weight* update, W_{ijk} is inversely proportional to the standard deviation. Thus, the normalized intra-weighting is defined as follows:

$$W_{ijk} = \frac{\frac{1}{\sigma_{ijk}}}{\sum_k \frac{1}{\sigma_{ijk}}}.$$

The *inter-weight* update is the part which directly relates to the relevance feedback. Let N_{RT} be the specified number of matching results returned from the system, and $RT = \{RT_l | l = 1, 2, ..., N_{RT}\}$ be the results. The MARS system allows user to

specify a score to each results RT_l:

$$Score_l = \begin{cases} 3 & \text{(highly relevant)} \\ 1 & \text{(relevant)} \\ 0 & \text{(no opinion)} \\ -1 & \text{(nonrelevant)} \\ -3 & \text{(highly nonrelevant)} \end{cases}$$

For each r_{ij}, let $RT^{ij} = \{RT_l^{ij} | l = 1, 2, ..., N_{RT}\}$ be the most similar N_{RT} images in terms of r_{ij}. After initializing the inter-weight $W_{ij} = 0$, we update W_{ij} by the following procedure:

$$\begin{aligned} W_{ij} &= \begin{cases} W_{ij} + Score_l, & \text{if } RT_l^{ij} \text{ is in } RT \\ W_{ij} + 0, & \text{if } RT_l^{ij} \text{ is not in } RT \end{cases} \\ l &= 1, 2, ..., N_{RT} \end{aligned}$$

After updating W_{ij}, if $W_{ij} < 0$, set it to 0. The final step is to make the sum of weights to be 1:

$$W_{ij} = \frac{W_{ij}}{\sum_{i,j} W_{ij}}.$$

An intuitive explanation of the above procedure is that "*the more the overlap of relevant objects between RT and* RT^{ij}*, the larger the value of* W_{ij}."

So far we have not discussed the convergence issue of the inter-weighting procedure:

- The RT is the query result which depends on all the weighting factors ($\{W_{ij}\}$);
- A weighting factor W_{ij} is derived according to the overlapping between RT and RT^{ij}.

To conduct the desired RT, the procedure includes a process to update $\{W_{ij}\}$ and RT iteratively. In most cases, it converges after a certain number of iterations. However, some cases may lead to a non-converging state. So, there is a parameter P_{fd} which denotes the maximum number of iterations to execute when the procedure

cannot determine the converged RT. According to the experiments, the converging ratio of $P_{fd} = 3$ is acceptable (though depending on the dataset).

2.1.3 Accuracy Measurements

In studying a CBIR system, an essential question is how well the system performs. Since an image query is often processed by similarity matching, difference/error based accuracy statistics may not suitable in this case. In this section, we describe two measurements which are widely accepted in the CBIR research community.

Precision and Recall

The first category of evaluation metrics are borrowed from information retrieval studies. The main tool is the *precision* and *recall* measures [14]. The fundamental definition is as follows: giving a data set D and a query q, the data set can be categorized into two sets. One is the set $R(q)$ which contains the instances relevant to the query, and the other is the complement set $\overline{R}(q)$ which contains irrelevant instances. Suppose the system produce a set of instances as the answer $A(q)$ in response to the query q. The precision is defined as the ratio between the number of the retrieved relevant instances and the number of the total retrieved instances (the answer):

$$p = \frac{|A(q) \cap R(q)|}{|A(q)|},$$

while the recall is defined as is the ratio between the number of the retrieved relevant instances and the number of the total relevant instances:

$$r = \frac{|A(q) \cap R(q)|}{|R(q)|}.$$

Although the two measures are useful, there are cases in which they are not sufficient to reflect the performance of a CBIR system. The first problem is that the selection of the relevant set in image retrieval is not as stable as that in conventional text retrieval. For a document, human judgment of whether a certain document is

relevant to a certain query is quite stable, and the judgment often strongly connects to certain elements in the document (e.g., keywords). In the context of image query, the judgment of "relevance" is blurred from person to person, and even from time to time. The second problem is that a CBIR system usually returns a ranked list of images. In an ideal system, a query result is an ordered list of the whole image database. However, in practical implementation, the query result is limited to a given size, say, *top-N* most similar images. When the number of relevant images is greater than N, the recall metric is meaningless.

In spite of the drawbacks, precision and recall can be useful if we define the relevant image sets for experiments. That is, we define a relevant test set $R'(q)$ and treat it as $R(q)$. By ignoring the possible mis-classified images $I = \{x \mid x \in R(q), x \notin R'(q)\}$, the first drawback can be ignored. Since we have $R'(q)$, we can easily require the system to return the answer list whose size $N \geq |R'(q)|$. Thus, the second drawback would not happen.

By implicitly assuming the conditions mentioned above, many researches use precision and recall analysis to evaluate a CBIR system. In such analysis, a defined relevant image set is called a *ground-truth* set. For a given ground-truth set $GT = \{x_i \mid i = 1..k\}$, the following condition is true:

$$R'(x_i) = GT, \forall x_i \in GT.$$

ANMRR

When performing the precision and recall analysis over a number of ground-truth sets with different sizes, a problem is how to "normalize" the effect of the chosen size of the top-N list. In searching for a suitable objective measure, we finally adopt the *Average Normalized Modified Retrieval Rank* (ANMRR) metric. The ANMRR is used in the MPEG-7 standardization process to quantitatively compare the retrieval accuracy of competing visual descriptors. This metric can be viewed as a modified combination of precision and recall metrics, and is a normalized index to rate the

overall query accuracy of a system. For a query image, this measurement favors a matched ground-truth result and penalizes a missing ground-truth. We briefly describe the formula of ANMRR in the following paragraphs. Details can be found in the references[16][17].

For a query image q with a ground-truth size of $NG(q)$, $rank(k)$ is the rank of the k-th ground-truth image on the top-N result list. Then,

$$\begin{aligned} Rank(k) &= \begin{cases} rank(k) & \text{if } rank(k) \leq K(q) \\ 1.25 \cdot K(q) & \text{if } rank(k) > K(q) \end{cases} \\ \text{where } K(q) &= \min\{4 \cdot NG(q), 2 \cdot \max[NG(q), \forall q]\}. \end{aligned}$$

Here we define a threshold $K(q)$. If the retrieval rank is larger than $K(q)$, we consider it a "miss" and penalize this result as rank $1.25 \cdot K(q)$. Based on the above definition, we compute the *Average Rank* (AVR) over the whole ground-truth set for query q as:

$$AVR(q) = \frac{1}{NG(q)} \sum_{k=1}^{NG(q)} Rank(k).$$

However, AVR has a drawback that the value depends on the size of the ground-truth set. For example, in the case of perfect match, the AVR is:

$$\begin{aligned} AVR_{perfect}(q) &= \tfrac{1}{NG(q)} \textstyle\sum_{k=1}^{NG(q)} Rank(k) \\ &= \tfrac{1}{NG(q)}(1 + 2 + 3 + ... + NG(q)) \\ &= \tfrac{1}{NG(q)} \cdot \tfrac{NG(q) \cdot (1 + NG(q))}{2} \\ &= \tfrac{1 + NG(q)}{2} \end{aligned}$$

That is, AVR would be very different for different ground-truth sets, even in perfectly matching cases. To eliminate influences of $NG(q)$, the *Modified Retrieval Rank* (MRR) is introduced:

$$MRR(q) = AVR(q) - 0.5 \cdot [1 + NG(q)].$$

The value of $MRR(q)$ is greater than or equal to 0, but the upper margin (the case of totally miss) still depends on $NG(q)$. This leads to the definition of the Normalized

Modified Retrieval Rank (NMRR):

$$NMRR(q) = \frac{MRR(q)}{1.25 \cdot K(q) - 0.5 \cdot [1 + NG(q)]}.$$

By the definitions above, the range of $NMRR(q)$ is $[0, 1]$. The value 0 indicates a perfect match that all the ground-truth pictures are included in the top-rank list. On the other hand, the value 1 means no match. Finally, we have the *Average Normalized Modified Retrieval Rank* (ANMRR) over the test cases:

$$ANMRR = \frac{1}{NQ} \sum_{q=1}^{NQ} NMRR(q),$$

where NQ is the number of queries.

2.2 Digital Rights Management

As the amount of digital content grows, the demand of digital rights management (DRM) becomes higher. In this section, we describe a simple model of DRM, and a few techniques used to protect digital contents.

2.2.1 Digital Media Lifetime

Digital rights management (DRM) is a concept that related technologies can restore the value of content. The words from NIST may be a formal definition of DRM:

> *"DRM is a system of IT components and services along with corresponding law, policies and business models which strive to distribute and control IP and its rights."*

The above statement can be interpreted as that a DRM system is a collection of components to support (legal-aspect) copyright issues. To model a DRM system in terms of technologies, a content-centric view (such as used in the MPEG-21 [18]) is that a digital content is accessed by a series of provider-consumer links. A complete DRM framework provides functionality to ensure that each end can only perform

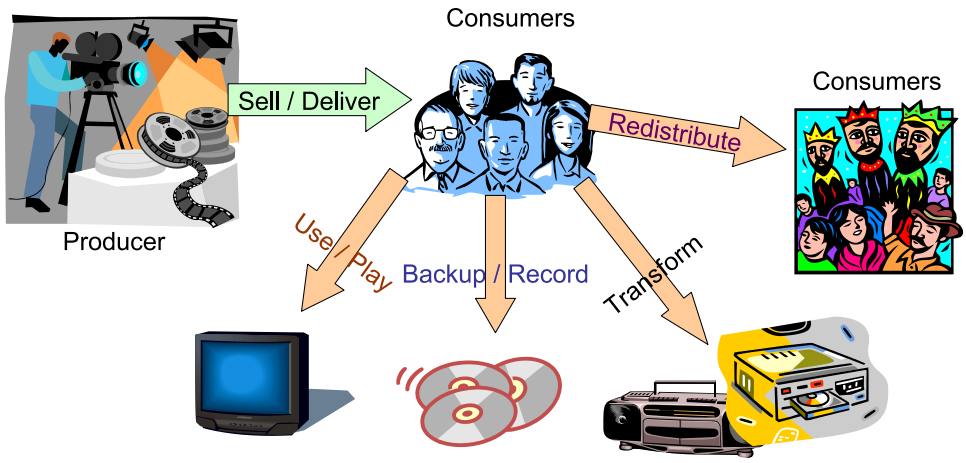


Figure 2.5: The life of a digital content.

legal access and usage operations. Figure 2.5 illustrates that a piece of digital content is produced by the producer, delivered to the consumers, used (including playback, backup, or transformation) by the consumers, and maybe re-distributed to other consumers.

According to the application scope, DRM technologies can be roughly divided into two categories. The first category protects a content from illegal accesses. If the delivery channel is interactive, the provider can authenticate the consumer, and decide whether to transmit the content or not. If the delivery channel is not interactive, such as broadcasting, the provider should always transmit the content. To prevent illegal accessing, the transmitted data should contain addressing information. In addition to the accesses from "normal" consumers, both schemes suffer from eavesdroppers in the middle of a link. Thus, cryptographic algorithms are often used to scramble the original content. Theoretically speaking, only the addressed consumer can restore the original content from the received data. Another restriction in access can be achieved by enforcing the use of a license. For example, an MPEG-21 REL [19] license describes the issuers and the grants. Each grant describes the principles, the rights, the resources, and the condition. A compliant device should determine the access/usage rights according to the license.

The second category is mainly used after the content is accessed. Some techniques are used to check the integrity of the content. For example, fragile watermarks are sensitive to small of alternation, and are often used to detect malicious modification of a content. Other techniques are used to identify the copyright of a content. For instance, robust watermarks can resist a certain distortion (such as transcoding or geometrical transformation) of the content, and can be used to identify the original copyright holder of a distorted content. Yet another techniques are used to identify the distributors of a content. For example, traitor-tracing methods can be used to identify the malicious distributors of a collaboratively hacked content.

2.2.2 The Conditional Access in DTV

In the MPEG-2 system standard, a three-layer encryption structure is constructed to protect the media data and to satisfy the requirements of long-term user password versus short-term (time-varying) control signaling [20]. A typical digital TV (DTV) conditional access receiving terminal is shown in Fig. 2.6. At the top layer, the so-called "Control Word" (CW) is the key that decrypts the contents. This key may change very frequently, say, once every a few seconds, so that a fast and simple decryption algorithm would not be broken easily. The control words are carried by an encrypted stream called ECM (Entitlement Control Message). To retrieve CW from the ECM stream, the Service Key (SK) must be decoded first. In addition to the User Key (UK), which may be contained in a Smart Card, the EMM (Entitlement Management Message) information is needed to decode SK. With the aid of EMM, the broadcaster can change the status of the user accessibility of contents. For example, the broadcaster can disable a user's access if the subscription is overdue. In short, other than the key hold by the user, there is a need of additional information sent by the broadcaster to decrypt the media content.

From the above discussions, we can see that a typical broadcast conditional access system encrypts digital content using more than one layer of keys. This is

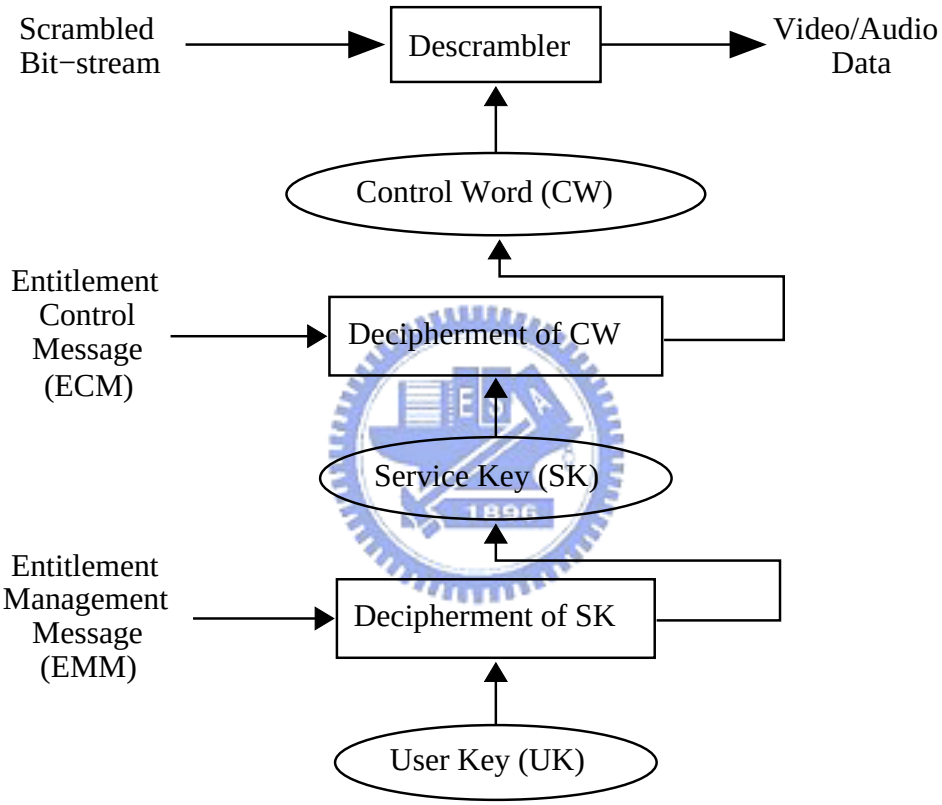


Figure 2.6: A typical MPEG-2 conditional access receiver.

an elegant design to meet both the complexity and security requirements. However, this design assumes that the key streams, similar to EMM and ECM, are sent by separate data packets and then synchronized with the contents and the User Key. This could increase quite a bit of overhead and complexity for layered bit streams. For example, to broadcast a protected content over Internet, we may send the key to users via a reliable channel (such as RTSP connection [21]), while the content goes through an unreliable channel (such as RTP sessions [22]). A reliable channel guarantees information correctness by sacrificing delivery speed, and it is likely that the key information is out-of-sync to the corresponding content.

2.2.3 A Watermarking Scheme

In this section, we describe the robust watermarking scheme used in our simulations in Chapter 4. The original paper [23] uses genetic algorithm (GA) to find the best mid-frequency coefficients for embedded watermark. The watermarking technique is a component in our example system to demonstrate the feasibility of our proposed architecture. Thus, we only give a brief description of the embedding and extraction algorithms below.

2.2.4 The Embedding Algorithm

Let the input image X with size $M \times N$. The goal is to embed a binary watermark W $(M_w \times N_w)$ into the image imperceptibly. The first step is to divide X into 8×8 blocks, and transform the image to the DCT domain. The transformed image Y is expressed by

$$Y = \bigcup_{m=1}^{M/8} \bigcup_{n=1}^{N/8} Y_{(m,n)},$$

where $Y_{(m,n)}$ denotes the 8×8 DCT block at the m-th row and the n-th column. The coefficients of a block can be expressed in a list of zig-zag scanned order:

$$Y_{(m,n)} = \bigcup_{k=0}^{63} Y_{(m,n)}^{(k)},$$

where $Y^{(k)}_{(m,n)}$ denotes the k-th DCT coefficient (in zig-zag scan order) of the block $Y_{(m,n)}$. To enhance the security level, the bits of the original watermark is permuted with a secret key_0:

$$W_P = \text{permute}(W, key_0).$$

To embed W into X, we need to select $64 \times M_w \times N_w/(M \times N)$ coefficients in each block $Y_{(m,n)}$. That is, W_P is also divided into $(M/8) \times (N/8)$ blocks, and each block contains $64 \times M_w \times N_w/(M \times N)$ bits. The original work uses GA to find the best solution, and the frequency set F records the selected frequencies for embedding. The mapping between the embedding frequencies and the coefficient index k is the second secret key_1. Next, the reference table $R = \{R(i)|i \in F\}$ is generated:

$$R(i) = \sum_{m=1}^{M/8} \sum_{n=1}^{N/8} \left(\frac{Y^{(0)}_{(m,n)}}{Y^{(i)}_{(m,n)}} \right), i \in [1, 63].$$

Then, the polarities are defined by

$$P^{(i)}_{(m,n)} = \begin{cases} 1 & \text{if } Y^{(i)}_{(m,n)} \times R(i) \geq Y^{(0)}_{(m,n)}, i \in F, \\ 0 & \text{otherwise;} \end{cases}$$

The watermark bits are embedded in to Y by the following rules:

$$Z^{(i)}_{(m,n)} = \begin{cases} Y^{(i)}_{(m,n)} & \text{if } P^{(i)}_{(m,n)} = 0 \text{ and } W^{(i)}_{P,(m,n)} = 0, i \in F, \\ Y^{(i)}_{(m,n)} + 1 & \text{if } P^{(i)}_{(m,n)} = 0 \text{ and } W^{(i)}_{P,(m,n)} = 1, i \in F, \\ Y^{(i)}_{(m,n)} & \text{if } P^{(i)}_{(m,n)} = 1 \text{ and } W^{(i)}_{P,(m,n)} = 1, i \in F, \\ Y^{(i)}_{(m,n)} - 1 & \text{if } P^{(i)}_{(m,n)} = 1 \text{ and } W^{(i)}_{P,(m,n)} = 0, i \in F. \end{cases}.$$

In the above equations, $W^{(i)}_{P,(m,n)}$ is the watermark bit (in W_P) which correspond to the i-th coefficient of block $Y_{(m,n)}$. The final step is to apply inverse DCT to Z to obtain the watermarked image X_c.

2.2.5 The Extraction Algorithm

The extraction process requires key_0 and key_1, and does not need the original image X. Assume the attacked image is X'', and its DCT transformed image is Y''. With

the key_1 and Y'', we can derive the estimated reference table R' by the same equation in the previous section. Then, the permuted watermark is extracted by the following formula:

$$W'^{(i)}_{P,(m,n)} = \begin{cases} 1 & \text{if } Y''^{(i)}_{(m,n)} \times R'(i) \geq Y''^{(0)}_{(m,n)} \forall i; \\ 0 & \text{otherwise;} \end{cases}$$

$$W'_P = \bigcup_{m=0}^{M/M_w-1} \bigcup_{n=0}^{N/N_w-1} W'^{(i)}_{P,(m,n)}, i \in F.$$

Finally, the we inversely permute with key_0 to get the extracted watermark:

$$W' = \text{inverse\_permute}(W'_P, key_0).$$