3.2 Design Goals

In a typical Query-by-Example (QBE) CBIR system with relevance feedback capability, the problem is how one utilizes multiple image features and multiple query instances (images) to derive the proper search parameters. Multiple features and multiple instances represent two different aspects. The former is how we describe an image in an application; the latter is how we guess the user intention using the given instances. There exist many proposals on combining multiple features for image search such as using Borda counts[25]. Methods of combining multiple instances are usually considered as a part of a relevance feedback function. There are several existing CBIR proposals containing relevance feedback such as MARS[15][26] and iPURE[27]. In many cases, the multi-instance analysis process uses the selected features. Since the feature selection problem is a design-time issue, the analysis method varies from application to application. For example, if the features are expressed as a vector of moments, the weighting factors for each moment can be computed by the boosting method[28], as described in [29] and [30].

In our previous project, we developed an MPEG-7 testbed [31] and thus have used it to examine several low-level MPEG-7 features. We observed that subjectively similar pictures tend to be close (near) in one or more feature spaces. Another observation is that a low-level feature often has (somewhat) different values when it is extracted from the same picture with different spatial resolutions and/or picture quality (SNR scalability). Our investigation finds that people often design a QBE system with feedback under the assumption that a sufficient number of query instances or feedback iterations can be provided by the user. However, this assumption is not always true in a real-world application[32][33]. Often, the sample size is very small (one to three) and the information contained in the samples may not be all consistent in terms of data clustering.

Based on our observations, we are motivated to develop a user perception estimation algorithm, which tries to make a correct conjecture on the user intention based on the small number of samples (instances) provided by the user. For simplicity and fast calculation, our system uses only low-level features for automatically high-volume feature extraction and matching. Another consideration is that it incorporates only simple distance-based weighting and matching scheme to make it easily integrated into various environments. We also consider a simple user interface for relevance feedback: the application only asks users to label positive or negative images in a free way. (No specific number of feedback images is required.)

3.3 Feature Weighting Method

There are several ways to combine different low-level features. Here we adopt a straightforward one: weighted sum of feature distances. We use the originally designated distance definition of individual features. Our focus is to find the most appropriate weight of each feature to produce an effective combined distance measure. Thus, our method preserves the individual feature space properties. In this scheme, the user perception is expressed by a weighting vector. Note that the weighting vector is derived from the multiple instances provided by the user.

Similar to many other image retrieval schemes, we assume the following conditions are satisfied:

- All the basic feature distance metrics are bounded.
- Two perceptually similar images have a small distance in at least one feature space.
- Low-level features are locally inferable[34]. That is, if all the feature values of two images are fairly close, then the two images are perceptually similar.

In addition to the above assumptions, we add another conjecture: if two images have a large distance value in a specific feature space, we cannot determine the perceptual similarity of them based merely on this feature. Note that this feature space is simply irrelevant to our perception. It does not necessarily decide dissimilarity in perception.

Different from several well-known CBIR systems, our system does not rely on *a priori* feature distributions. These distributions may help to optimize inter-feature normalization, as in MARS[15], to produce better matching performance. However, they often introduce overheads and require high computation. Even if feature distributions are available, they may not lead to appropriate normalization. More importantly, user perceptions do not necessarily match the feature distributions in the database. Thus, we try to design our method to be independent of feature distributions as shown below. The need of normalization is eliminated because of the way we define distance function.

In summary, our feature weighting and combination principle is: *given two user-input query images, if they are farther apart in a certain feature space, this feature is less important in deciding the perceptual similarity for this particular query.* Suppose we have a query image set with n samples, $Q = \{q_i \mid i = 1..n\}$, and an available basic feature set $F = \{F_j \mid j = 1..m\}$. Let f_{ij} denote the value of feature F_j for image q_i. The normalized distance function for feature F_j is $d_j(f_{1j}, f_{2j}) = n_j * D_j(f_{1j}, f_{2j})$, where $D_j(f_{1j}, f_{2j})$ is the designated distance function for F_j, and n_j is the normalization factor for F_j, which sets the normalized value $d_j(f_{1j}, f_{2j})$ in the range of $[0, 1]$. Though n_j is an *a priori* information, we will see that it can be safely discarded at the end of this section.

To measure the sparseness of a set of feature points, first we assume all the feature distances satisfy triangular inequality, and there exists at least one hyper-sphere, inside which all the feature points are located. A hyper-sphere can be defined by a "pivot" (centroid) and a radius. Among all possible spheres in the F_j space, we call the smallest one as the bounding sphere, and its radius is defined as the scatter number of this space.

Based on the above discussion, we define the scatter number (s_j) of Q for feature

F_j as follows:

$$s_j = \begin{cases} 1, & \text{if } \mid Q \mid = 1 \quad (1) \\ \frac{1}{2} d_j(q_1, q_2), & \text{if } \mid Q \mid = 2 \quad (2) \\ max_{\forall k} r_j^{(k)}, & \text{if } \mid Q \mid \geq 3 \quad (3) \end{cases},$$

where $\mid Q \mid$ is the size of set Q.

In condition (1), because we do not have enough information to determine the scatter of each feature, we simply assign a default value (= 1) to s_j. In case (2), we only have two query samples $Q = q_1, q_2$. Thus, the minimal bounding radius is half of the distance between them. In case (3), we have more than two query instances, and we can derive the bounding radius using geometry theorems. Let $Q^{(k)} = \{q_1^{(k)}, q_2^{(k)}, q_3^{(k)}\}$ be the k-th combination out of the total C_3^n combinations of the query set Q, and they satisfy the following criterion:

$$\begin{array}{c} t_1^{(k)} = d_j(q_1^{(k)}, q_2^{(k)}) \\ t_2^{(k)} = d_j(q_2^{(k)}, q_3^{(k)}) \\ t_3^{(k)} = d_j(q_3^{(k)}, q_1^{(k)}) \\ t_1^{(k)} \geq t_2^{(k)} \geq t_3^{(k)} \end{array},$$

Under this condition, there are two sub-cases: one is $(t_1^{(k)})^2 \geq (t_2^{(k)})^2 + (t_3^{(k)})^2$, and the other is $(t_1^{(k)})^2 < (t_2^{(k)})^2 + (t_3^{(k)})^2$. When the former one occurs, the bounding radius is $r_j^{(k)} = \frac{1}{2} t_1^{(k)}$; when the latter one occurs, $r_j^{(k)}$ is the solution of the two equations:

$$(t_1^{(k)})^2 = (t_2^{(k)})^2 + (t_3^{(k)})^2 - 2(t_2^{(k)})(t_3^{(k)}) \cos\theta$$
$$(t_1^{(k)})^2 = (r_j^{(k)})^2 + (r_j^{(k)})^2 - 2(r_j^{(k)})(r_j^{(k)}) \cos 2\theta$$

The scatter numbers may be interpreted as an "importance" indicator of that feature, because a larger bounding sphere means that feature points are spreaded over a large region. Based on the previously described principles, we give less *perception weight* to a more scattered feature (F_j):

$$w_j = \frac{1}{s_j} * \left(\sum_{k=1}^{m} \frac{1}{s_k} \right)^{-1}.$$

The distance function (of two images, q_1 and q_2) combining m features is then defined as

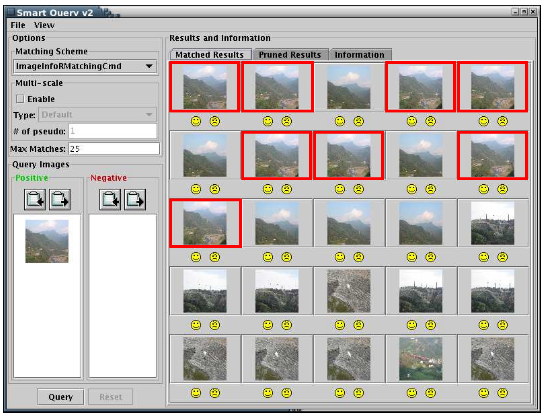
$$D(q_1, q_2) = \sum_{j=1}^{m} w_j * d_j(f_{1j}, f_{2j}).$$

Finally, the distance function between image I and n query instances (Q) is defined by

$$D(I, Q) = \min_{i=1..n} D(I, q_i).$$

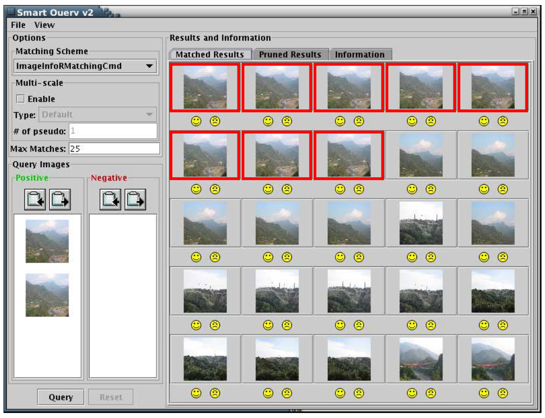
Note that the normalization factor n_j is canceled in every $w_j * d_j(f_{1j}, f_{2j})$ term. This implies that we can safely ignore the distance normalization problem as long as all the feature metrics are bounded.

To test the efficiency of the method, we perform subjective queries against the image database that will be described in detail in Sec. 3.7.1. Three image global features defined by MPEG-7[24] are adopted. They are scalable color, color layout, and edge histogram. Figure 3.4 illustrates how our proposed method improves the query accuracy. Figure 3.4(a) shows the results when a user specifies one image as the query example. Since we do not have enough information to weight the features, we simply assign them equal weights. The top-25 results are listed from left to right and top to bottom, with the most similar at the top-left corner. The red-boxed images are the ground-truth images. We may see that three non-ground-truth images are considered more similar than the ground-truth images. Figure 3.4(b) shows the results when a user gives two of the ground-truth images as the query examples. The system derives weighting factors for each feature, and the results are improved. All the ground-truth images occupy the top ranks, a desired result. Figure 3.5 shows the two groud-truth sets which occupies top ranks of Fig. 3.4(a). The top-left image of Fig. 3.5(a) is the query instance, and the second set (Fig. 3.5(b)) interferes the query results in the condition of Fig. 3.4(a).



Matching Scheme: ImageInfoRMatchingCmd
Multi-scale: Enable
Type: Default
# of pseudo: 1
Max Matches: 25

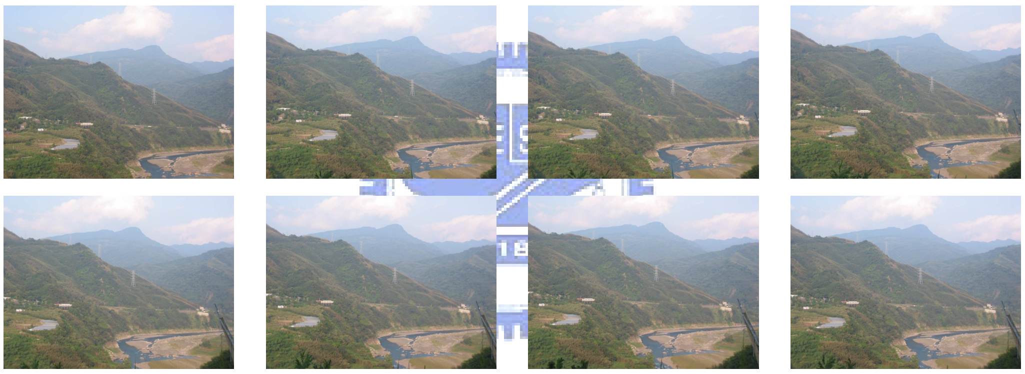
(a)



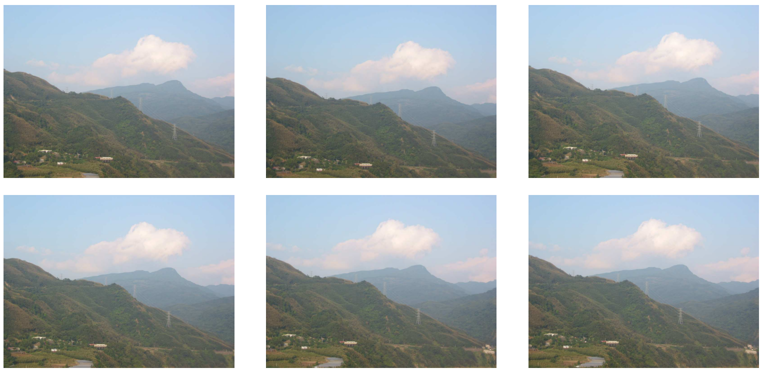
Matching Scheme: ImageInfoRMatchingCmd
Multi-scale: Enable
Type: Default
# of pseudo: 1
Max Matches: 25

(b)

Figure 3.4: Subjective result of multi-instance effects.



(a) Ground-truth set I



(b) Ground-truth set II

Figure 3.5: The two interfering groud-truth sets in Fig. 3.4(a).

3.4 Negative Instances

In this section, we will describe how to use irrelevant (negative feedback) images to improve the query accuracy. For a typical QBE search, a user provides a non-empty set of relevant (positive feedback) images. Suppose we can also ask the user to select negative (irrelevant) images. In the following proposed scheme, negative images are not included in computing the perceptual weights. This is due to the following observations.

1. *All positive examples are alike; each negative example is negative in its own way*[33].
2. The human perception of similarity and dissimilarity may not be (linearly) additive.
3. When an image is considered dissimilar to the query one, we do not know which features (one or many) dominate in producing the perceptual dissimilarity.

So we use negative images in the following way: they create "holes" in the feature space. That is, the database images located inside the pruning radius and close to the negative images are removed from the top-N (similar) list. As shown in Fig. 3.6, a negative sample is denoted as g_i and the positive samples are denoted as p_1, p_2, and p_3. Essentially, we conduct a pruning process for removing positively correlated images based on the given negative image(s). Let Q_p and Q_n are the positive and the negative image sets respectively. A *pruning radius* associated with a negative image $g_i \in Q_n$ is specified by $r_p(g_i) = D(g_i, Q_p)$. An image I_r is thus removed from the top-N list if I_r is located in a pruning region:

$$\exists g_i \in Q_n \text{ satisfies } \begin{cases} D(I_r, g_i) < r_p(g_i) \\ D(I_r, Q_p) > D(I_r, g_i) \end{cases} .$$

There are two conditions given in the preceding equation. An intuitive explanation to the second condition is that if an image is closer to Q_n than Q_p, it is excluded

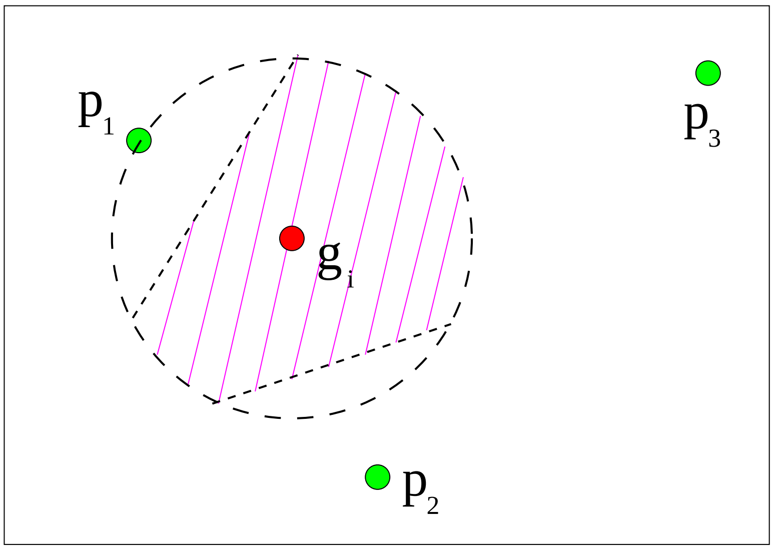
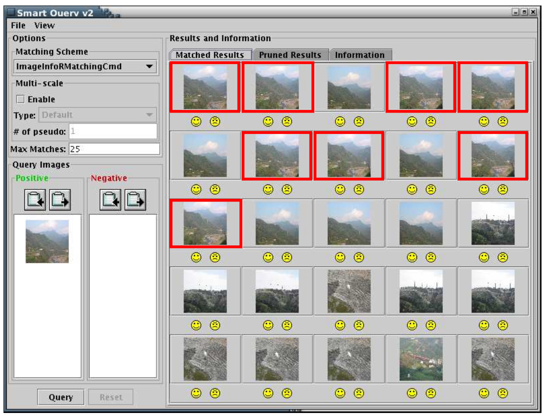


Figure 3.6: Pruning area in the combined feature space.

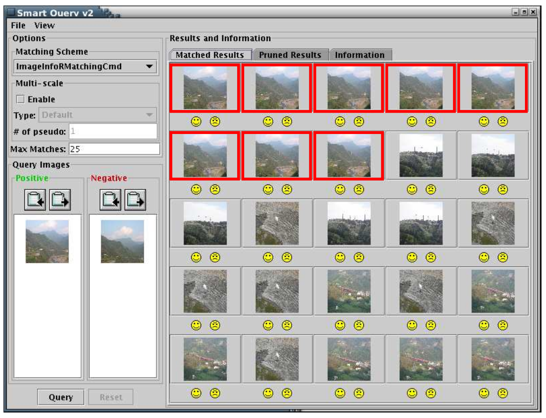
from the top-N list. Since the negative feedback sample set is small and incomplete, we do not want to exclude the images that are a bit far away from both sets but are slightly closer to a negative sample. Therefore, the first condition gives a maximum pruning radius. Thus, our pruning operation starts from the highest priority item on the top-N list. If an image is closer to Q_n than Q_p and is located inside the pruning radius, it is excluded from the top-N list.

Figure 3.7 illustrates the matching results with and without the negative query instance. Figure 3.7(a) is the query results of using a single positive instance. After assigning the highest-ranked non-ground-truth image as the negative feedback, the query results is shown as Fig. 3.7(b). We may see that the query accuracy is improved even when using equal-weighted combined distance function (remember that only positive instances participate in the weighting estimation).



Matching Scheme: ImageInfoRMatchingCmd
Multi-scale: Enable
Type: Default
# of pseudo: 1
Max Matches: 25

(a)



Matching Scheme: ImageInfoRMatchingCmd
Multi-scale: Enable
Type: Default
# of pseudo: 1
Max Matches: 25

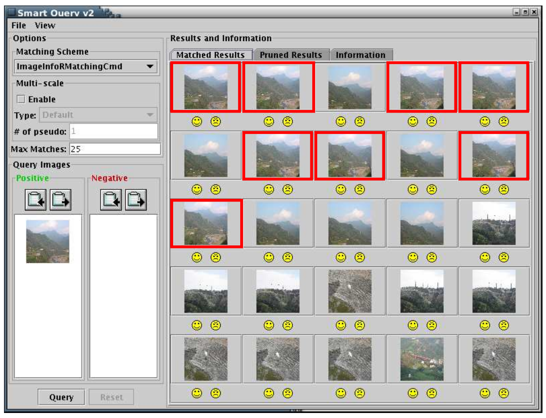
(b)

Figure 3.7: Subjective result of negative instance effects.

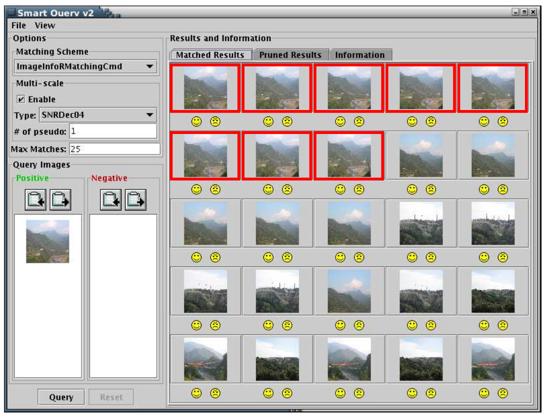
3.5 Pseudo Instances

In the case that the number of query images is too small, we use the multi-scale technique to create pseudo query images. The term "scale" here refers to either the spatial resolution or the SNR quality. It is based on the conjecture that the down-sampled or noise-added images are subjectively similar to the original version. We also observe that a low-level feature often have somewhat different values at different scales.

An unstable (sensitive) feature in our definition yields a large distance value among the scaled images derived from the original with different scales. The measure of instability is again specified by the scatter number s_j defined in Sec. 3.3. Stable features often represent the most noticeable features of an image and they in term are often the features that the inquiring users desire. Therefore, we come up with another principle: *We give the stable features of a query image more confidence (more weight) in searching for its similar images.* Thus, we include these pseudo images into the query set. The combined procedure thus puts less weight to more scattered features, which may be due to either perceptual irrelevance or feature instability. Two possible pseudo image generation methods are described later in Sec. 3.7.1. We will see that the pseudo image improves accuracy when the number of



Matching Scheme: ImageInfoRMatchingCmd
Multi-scale: Enable
Type: Default
# of pseudo: 1
Max Matches: 25



Matching Scheme: ImageInfoRMatchingCmd
Multi-scale: ✓ Enable
Type: SNRDec04
# of pseudo: 1
Max Matches: 25

(a) (b)

Figure 3.8: Subjective result of pseudo instance effects.

input images is one or two. Hence, the feature stability principle is justified mostly by observations and experiments.

The effect of pseudo-image generation is illustrated by Fig. 3.8. As before, Fig. 3.8(a) is the single positive image query. When we enable the multi-scale pseudo-image generation (one SNR pseudo image in this example), the query returns the desired result, as shown in Fig. 3.8(b). Often, by incorporating pseudo image concepts, the system gives users the best results at the first query iteration.

3.6 Architecture

The proposed CBIR query system architecture is summarized by Fig. 3.9. The original positive query (input) images are used to generate pseudo-images. Together they form the query set. The query set is fed into the user perception analysis process to estimate the weighting factors. Then, the query set and the weighting factors are passed to the image matching process to compute image similarity. A tentative matching list is thus produced. Then, the pruning process based on the supplied negative images is applied to the tentative matching list and some “irrelevant” images may be removed. At the end, we receive the final top-N list.

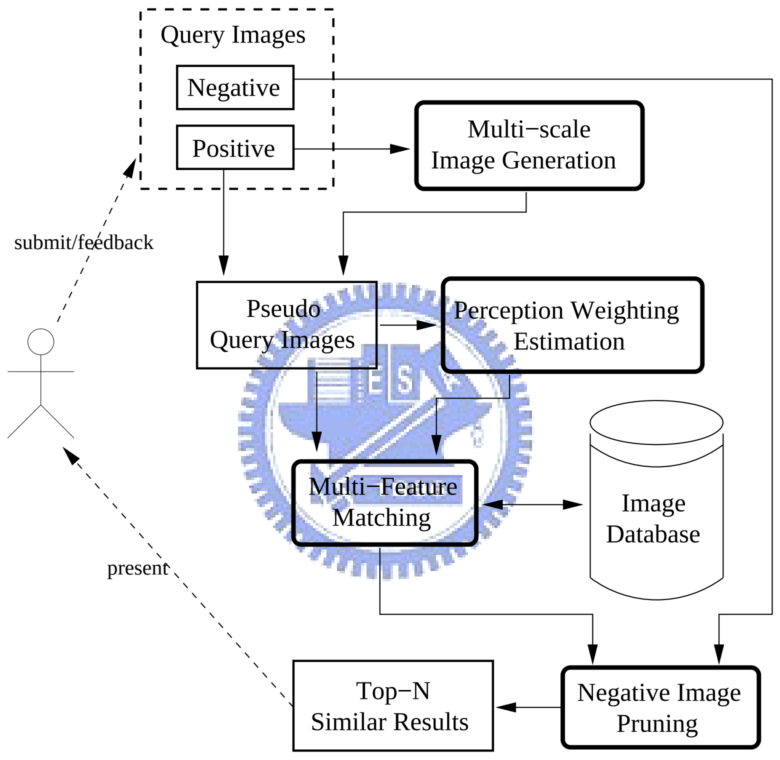


Figure 3.9: Proposed perception estimation and query system.