# Decision-Theoretic Approaches to Display Strategies in Content Based Image Retrieval

Georgios Stefanou Simon P. Wilson

**Abstract**

In this report we continue development of a Bayesian CBIR system by considering the issue of an image display strategy. This is a decision problem and so within the Bayesian paradigm is to be solved by decision theory. We show how different display strategies may be quantified by an appropriate utility function, and compare different optimisation strategies. Examples are given.

## 1 Introduction

In previous work we have described our extensions to the Bayesian content based image retrieval (CBIR) system *PicHunter* of [1]. In this short report we discuss how the Bayesian paradigm may be extended to another aspect of the relevance feedback process, namely the decision as to which images to display at each iteration. This is a decision problem, and as such it is to be solved within the Bayesian paradigm by the methods of decision theory.

This report is organised as follows. In Section 2 we describe the Bayesian learning algorithm for CBIR. In Section 3 we describe the decision theory solution to the display strategy problem. Section 4 concludes with some examples.

## 2 A Brief Description of a Bayesian CBIR system

Our approach is an extension of [1]. We consider a database of images $\mathcal{I} = \{T_1, \ldots, T_N\}$. The objective is to determine the "target" image $T \in \mathcal{I}$ that the user requires. $T$ could be a specific known image in the database or more generally that image in $\mathcal{I}$ which best satisfies the user's subjective search criteria. The determination of $T$ is accomplished by displaying a set of $N_D$ images from $\mathcal{I}$, from which the user picks one that best satisfies what is being looked for. The system uses this information to select another image set, from which the user picks one, and so on. We define $D_i \subseteq \mathcal{I}$ to be the set of displayed images at the $i$th iteration of this process, and $A_i \in D_i$ to be the image picked, also known as the user action. We define $H_t = \{D_1, A_1, D_2, A_2, \ldots, D_t, A_t\}$ to be the history of displayed images and user actions up to the $t$th iteration.

The learning algorithm is based around the user model for the probability of which image a user picks from $D_k$:

$$P(A_k \,|\, D_k, T = T_i, \sigma, F) \;=\; \frac{\exp\left(-d_F(A_k, T_i)/\sigma\right)}{\sum_{T_j \in D_k} \exp\left(-d_F(T_j, T_i)/\sigma\right)}, \tag{1}$$

where $\sigma$ is a precision parameter and $d_F$ is a normalised distance measure in the set of image features $F$. In this case we have 3 sets of features: global colour, texture and segmentation features, so $F \in \{\text{GC, TX, SG}\}$.

The unknowns are $T$, the precision parameter $\sigma$ and the feature set $F$. Given $H_t$, our knowledge about these unknowns is given by the posterior distribution:

$$P(T, \sigma, F \mid H_t) \;\propto\; \left( \prod_{k=1}^{t} P(A_k \mid D_k, T, \sigma, F) \right) \; P(T)\, P(\sigma)\, P(F), \tag{2}$$

where $P(T)$, $P(\sigma)$ and $P(F)$ are prior distributions that we assume are uniform: $P(T = T_i) = N^{-1}, i = 1, \ldots, N$, $P(\sigma) = 1, 0 \leq \sigma \leq 1$ and $P(F) = 1/3, F \in \{\text{GC,TX,SG}\}$.

Of interest in this report is the marginal posterior distribution of $T$:

$$P(T = T_i \mid H_t) \;=\; \int_0^1 \sum_{F \in \{\text{GC,TX,SG}\}} P(T_i, \sigma, F \mid H_t), \; i = 1, \ldots, N. \tag{3}$$

## 3 Deciding the Next Display Set $\mathbf{D_{t+1}}$

The question that this report addresses is the following. Based on $P(T = T_i \mid H_t)$, which set of images $D_{t+1}$ should be displayed next? This is a decision problem — we must decide which subset of $\mathcal{I}$ of size $N_D$ to display — and within the Bayesian paradigm, such problems are solved by decision theory.

We define a utility $U(D, T)$ that is the "worth" of picking the set $D$ to display when the target image is $T$. Since $T$ is unknown, we compute for each possible $D$ the expected utility with respect to $P(T = T_i \mid H_t)$:

$$\mathcal{U}(D) = \sum_{i=1}^{N} U(D, T_i)\, P(T = T_i \mid H_t). \tag{4}$$

The optimal set to display is that $D$ which maximises expected utility:

$$D_{t+1} = \arg \max_{\substack{D \subseteq \mathcal{I} \\ |D| = N_D}} \mathcal{U}(D). \tag{5}$$

### 3.1 The Most Probable Display Scheme

The most obvious display scheme is to display those $N_D$ images with the highest posterior probability. We observe that if we define

$$U_I(D, T) = \begin{cases} 1, & \text{if } T \in D, \\ 0, & \text{otherwise,} \end{cases}$$

then

$$\mathcal{U}_I(D) = \sum_{T_i \in D} P(T = T_i \mid H_t)$$

which is clearly maximised by those images with highest probability. We call this the indicator utility.

### 3.2 Other Display Strategies

A property of the most probable display scheme is that it tends to quickly display images in a small region of the feature space, clustered about the user actions, and ignores all images outside it. While this may be ultimately what is needed during a query, it may be more worthwhile to display images that maximise information to the system, at least in the early stages of the query process. We propose 2 utilities to model this idea.

#### 3.2.1 Variance Utility

We display a set of images that are widely dispersed in feature space. We can use the variance of the distances between images in $D$ and $T$ to define a measure of dispersion, thus

$$U_V(D,T) = \frac{1}{N_D - 1} \sum_{T_i \in D} (d(T_i,T) - \overline{d})^2,$$

where $d(T_i,T)$ is a normalised distance measure in feature space and

$$\overline{d} = \sum_{T_i \in D} d(T_i,T)/N_D$$

is the mean distance of images in $D$ to $T$.

#### 3.2.2 Entropy Utility

A measure of information is the reduction in entropy in the distribution of $T$ by selecting a particular display set. So we can define a utility based on the negative expected entropy of the posterior of $T$ from picking an image in $D$, expectation over the images in $D$:

$$U_E(D,T) = -\sum_{A_j \in D} \mathcal{E}(A_j,D)\, P(A_j \,|\, D,T), \tag{6}$$

where

$$\mathcal{E}(A_j,D) = -\sum_{i=1}^{N} P(T = T_i \,|A_j,D)\, \log(P(T = T_i \,|A_j,D))$$

is the entropy of the posterior distribution of $T$ given that $A_j$ is picked from $D$ (following Equations 2 and 3) and

$$P(A_j \,|\, D,T) = \frac{\exp\left(-d(A_j,T)/\overline{\sigma}\right)}{\sum_{T_j \in D} \exp\left(-d(T_j,T)/\overline{\sigma}\right)}$$

is the likelihood term as in Equation 1 but using the distance measure $d$ over the entire feature space and $\overline{\sigma}$ is the posterior mean of $\sigma$.

### 3.3 Optimisation Methods

Because $\mathcal{U}(D)$ is separable in each element of $D$, the optimal $D$ for the indicator utility can be easily computed. This is not the case if one moves to using the variance or entropy utilities. Evaluation of the expected utility for all possible $D$ is not an option as the number is large i.e. for $N = 1000$ and $N_D = 6$ we have about $1.37 \times 10^{15}$ possible subsets. For these, we have to resort to methods that are not guaranteed to find the optimal. We propose two Monte Carlo optimisation schemes.

#### 3.3.1 Random Generation

We randomly generate without replacement $K$ subsets $D^1, \ldots, D^K$. Then we let

$$D_{t+1} = \arg\max_{k=1}^{K} \mathcal{U}(D^k). \tag{7}$$

Each element of a set $D$ can be simulated from any distribution on $\mathcal{I}$; obvious choices are the uniform and $P(T \,|\, H_t)$. In this paper we choose the latter.

### 3.3.2 Simulated Annealing

For simulated annealing, we define a “neighbour” of a subset $D$ to be another subset with one different image. A simulated annealing algorithm then runs as follows:

1. Define an initial temperature $T_0$, a final temperature $T_{\min}$ and a cooling schedule $T_1, T_2, \ldots$. Randomly generate without replacement a set $D^0$, using $P(T \mid H_t)$. Let $k = 0$.
2. While $T_k > T_{\min}$
   - $k = k + 1$.
   - Select at random one image in $D^{k-1}$ and replace with another image in $\mathcal{I} - D^{k-1}$, randomly generated according to $P(T \mid H_t)$. Call this new set $D^{\text{new}}$.
   - With probability $\min\{1, \exp((\mathcal{U}(D^{\text{new}}) - \mathcal{U}(D^{k-1}))/T_k)\}$, let $D^k = D^{\text{new}}$ else $D^k = D^{k-1}$.
3. $D_{t+1} = D^k$.

Initial and final temperatures were decided on by using the methods of [2] and [3]. We looked at several different cooling schedules, and found that inverse linear ($T_k = a/(1 + bk)$) performed best. The choice of $P(T \mid H_t)$ to generate $D^0$ and $D^{\text{new}}$ can be changed, to for example the uniform, but we found that the method was not particularly sensitive to this choice. Finally, our definition of neighbour can be made more or less strict, by for example allowing two changes for $D^{\text{new}}$ or, conversely, only favouring replacement of one image in $D^{\text{new}}$ that is close in feature space to that image replaced. However we found that our choice was a compromise between a too small and too large change that offered a good accept rate.

Finally we note that computation time is limited in a live implementation of either optimisation scheme, so typically we can compute only a small number of expected utilities.

## 4 Examples

As an illustration of the method, we have a simulated database of only $N = 15$ images, each with only 2 features, for which queries are implemented by displaying $N_D = 3$ images. This is clearly an unrealistically small example but it has the advantages of allowing us to display what happens in feature space and, since there are only 455 possible subsets of size 3, to compute the exact optimal subset under all 3 utilities and compare with the results obtained by random sampling and simulation.

Figure 1 shows an example of the system where one image $A_1$ is picked from an initial display set $D_1$, and the resulting choice of $D_2$ according to the 3 utilities. Upper left of the figure are the 15 images in feature space with $D_1 = \{T_2, T_3, T_5\}$ highlighted. Image 3 is selected. We then see that, under $U_I$, we have $D_2 = \{T_3, T_8, T_{12}\}$, that is images close to that selected. For $U_V$ we have $D_2 = \{T_6, T_7, T_{13}\}$ and for $U_E$ we have $D_2 = \{T_7, T_{13}, T_{14}\}$, that is images that are widely separated in feature space are chosen. To explore the effectiveness of the optimisation methods, we repeated this experiment 1000 times, computing the optimal $D_2$ according to the random generation method and simulated annealing. For the random subset generation, we simulated $K = 100$ subsets. For the simulation annealing we used an inverse linear cooling schedule $T_k = a/(1 + bk)$ with $a$ and $b$ chosen so that the final temperature was reached in 100 iterations, thus both methods took the same time to compute. The results are compared with the exact calculation in Table 1 and we see that both non-exact methods are sub-optimal, but nevertheless do manage on average to find subsets with expected utility close to the optimal. Random generation appears to do slightly better than simulated annealing.

Finally, we move to the BAL database. In this case we cannot enumerate all possible subsets and so $D_2$ under the variance and entropy utilities is computed by the two optimisation schemes only. Table 2 compares the optimisation methods over 10 runs for an example from this database where $D_1$ consists of 6 images; note that we only have the exact result for the indicator utility. From the results for the

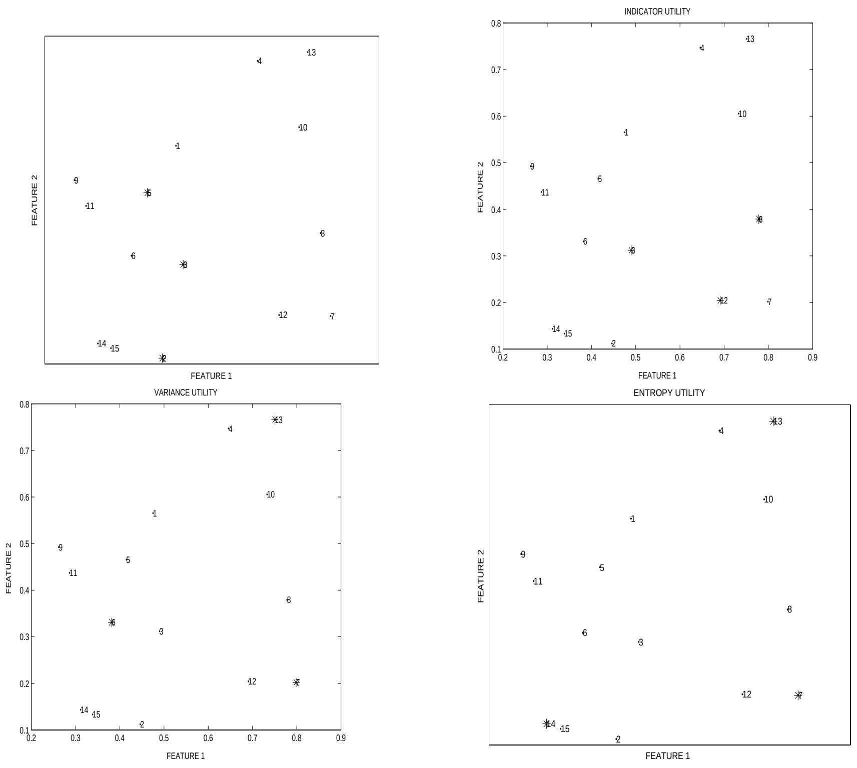

Figure 1: Feature space plots for the selection of $D_2$ for a simple database of 15 images given $D_1 = \{T_2, T_3, T_5\}$ and $A_1 = T_3$. Images in $D_2$ are highlighted by $*$.

| Utility | Exact Computation over all subsets | Random Generation of 100 subsets | Simulated Annealing 100 iterations |
|---|---|---|---|
| Indicator | 0.2643 | 0.2624 | 0.2591 |
| Variance | 0.2299 | 0.2264 | 0.2246 |
| Entropy | -2.6869 | -2.6879 | -2.6883 |

Table 1: The average of the expected utility for $D_2$ over 1000 runs for the 3 utility functions and three computation methods. All runs use the example of Figure 1.

| Utility | Exact Computation over all subsets | Random Generation of 100 subsets | Simulated Annealing 100 iterations |
|---|---|---|---|
| Indicator | 0.0157 | 0.0106 | 0.0100 |
| Variance | — | 0.7767 | 0.7583 |
| Entropy | — | -6.9642 | -6.9647 |

Table 2: The average of the expected utility for $D_2$ over 10 runs for the 3 utility functions and three computation methods using the BAL database.

indicator utility it appears that both optimisation methods can be significantly sub-optimal. From all 3 utilities it appears that random generation performs better than simulated annealing.

## 5 Conclusion

We have described a decision-theoretic approach to the problem of display set strategy in content-based image retrieval systems. The notion of utility is, we believe, a useful and intuitive way to quantify display strategy objectives. One is free to define any utility function at all, as long as computational issues can be successfully addressed.

It remains to say that the two new utilities that we have proposed — variance and entropy — are primarily of use in the early stages of a query, when the objective is to learn as much as possible about the user's target. Ultimately, one will want to resort to a utility that displays images close to the target, such as the indicator. An obvious way to do this is to consider a utility that is a convex weighted combination of the indicator utility with one of the other two, with the weight on the indicator utility increasing to 1 with the iteration, for example at the $t$th display set:

$$U(D,T) = \alpha_t U_I(D,T) + (1-\alpha_t)U_E(D,T),$$

with $0 \leq \alpha_t \leq 1$ and $\alpha_t \to 1$, and the entropy utility normalised from that in Equation 6 so that it lies in $[0,1]$ like $U_I(D,T)$. This is the subject of current work.

## References

[1] I. J. Cox, M. L. Miller, T. P. Minka, T. V. Papathomas, and P. N. Yianilos. The Bayesian image retrieval system, PicHunter: theory, implementation and psychophysical experiments. *IEEE Transactions on Image Processing*, 9:20–37, 2000.

[2] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[3] M. Lundy and A. Mees. Convergence of an annealing algorithm. *Mathematical Programming*, 34:111–124, 1986.