Chapter 7

Short-term learning

There are various reasons to doubt that any retrieval system (no matter how sophisticated) will always be able to find the desired images in response to the first query from a user. This is due to 1) the difficulty of the image understanding problem and 2) the fact that users themselves are not always sure of what they want to retrieve before browsing through the database. In practice, it means that retrieval is always an interactive process, consisting of the following sequence of events: 1) user provides a query, 2) system retrieves best matches, 3) user selects a new query, 4) process is iterated.

The interactive nature of the retrieval problem can be both a blessing and a curse for retrieval systems. On one hand, feedback provided by the user can be exploited to guide the search. This is a major departure from traditional vision problems and makes it feasible to build effective systems without solving the complete artificial intelligence problem [132]. In this context, a retrieval system is nothing more than an interface between an intelligent high-level system (the user's brain) that can perform amazing feats in terms of visual interpretation but is limited in speed, and a low-level system (the computer) that has very limited visual abilities but can perform low-level operations very efficiently.

On the other hand, users tend to be frustrated if the system does not appear to know how to integrate their feedback. This means that the low-level retrieval system cannot be completely dumb and, at the very least, should be able to *integrate* the information provided by the user throughout the entire course of interaction. Otherwise, it will simply keep oscillating between the image classes that best satisfy the latest query and convergence to the right solution will be difficult.

Consequently, in addition to a powerful image representation, a good retrieval system must also incorporate inference mechanisms to facilitate the user-interaction. However, the two problems cannot be solved in isolation, as the careless selection of the representation will make inference more difficult and vice-versa. In the previous chapters, we have established that probabilistic retrieval is a powerful solution to the problem of evaluating image similarity. We now show that it is also the natural answer to the inference problem.

7.1 Prior work

The design of inference algorithms is particularly difficult for retrieval systems based on holistic image similarity because, in this case, two different tasks must be accomplished. First, the system must figure out what exactly is the set of visual image properties or concepts that the user is interested in. Finding a good match for these concepts is possible only after they are identified. As the example in Figure 6.1 demonstrates, the first step cannot be accomplished from the observation of a single image, and several iterations of the interaction between user and retrieval system must occur before the latter knows exactly what the former is looking for, assuming that this is ever clear. By avoiding this first learning step, systems relying on localized feedback need to concentrate only on the second problem, which has easier solution.

Given this observation, it is somewhat surprising to realize that, while various solutions have been presented to the inference problem (commonly referred to as *relevance feedback* in the retrieval literature) [12, 32, 110, 150, 175], most of them are intimately related with image representations that preclude local similarity. In fact, to the best of our knowledge, only the "Four eyes" system [110] combines learning with local queries, and even these are restricted to image patches of a sizeable dimension.

With respect to the inference mechanisms, both the "Four eyes" [110] and "PicHunter" [32] systems are Bayesian in spirit. "Four eyes" pre-segments all the images in the database, and groups all the resulting regions. Learning consists of finding the groupings that maximize the product of the number of examples provided by the user with a *prior grouping weight*. This can be seen as an approximation to Bayes rule. "PicHunter" defines a set of actions that a user may take and, given the images retrieved at a given point, tries to estimate the probabilities of the actions the user will take next. Upon observation of these actions, Bayes rule gives the probability of each image in the database being the desired target.

The main limitations of these two systems are due to the fact that the underlying image representations and similarity criteria are not conducive to learning per se. For example, because there is no easy way to define priors for region groupings, in [110] this is done through a greedy algorithm based on heuristics that are not always easy to justify or guaranteed to lead to an interesting solution. On the other hand, because user modeling is a difficult task, [32] relies on several simplifying assumptions and heuristics to estimate action probabilities. These estimates can only be obtained through an ad-hoc function of image similarity which is hard to believe valid for all or even most of the users the system will encounter. Indeed it is not even clear that such a function can be derived when the action set becomes more complicated than that supported by the simple interface of "PicHunter". For example, in the context of local queries, the action set would have to account for all possible segmentations of the query image, which are not even defined a priori.

All these problems are eliminated by the Bayesian formulation of the retrieval problem introduced in this thesis because it grounds all inferences directly on the image observations selected by the user. In this chapter, we show that, by combining a probabilistic criteria for image similarity with a generative model for image representation, there is no need for heuristic algorithms to learn priors or heuristic functions relating image similarity and the belief that a given image is the target. Under the new formulation, 1) the similarity function is, by definition, this belief and 2) prior learning follows naturally from belief propagation according to the laws of probability [127, 75, 85, 76]. Since all the necessary beliefs are an automatic outcome of the similarity evaluation and all previous interaction can be summarized by a small set of prior probabilities, this belief propagation is very simple, intuitive, and extremely efficient from the points of view of computation and storage.

7.2 Bayesian relevance feedback

Following Cox et al. [32], we identify two types of searches: *target search* and *open-ended browsing*. While in target search users seek to find an image from a specific image class, in open-ended browsing they only have a vague idea of what they are looking for. It is relatively easy to extend the Bayesian retrieval model so that it can account for both situations. Instead of a single query[1] $\mathbf{x}$, we consider a sequence of queries $\mathbf{x}_1^t = \{\mathbf{x}_i\}_{i=1}^t$ and, instead of a class indicator variable Y, we define a collection $Y_1^t = \{Y_i\}_{i=1}^t$, where t is the iteration number. The event $Y_t = i$ indicates that the i^{th} image class is the target for iteration t.

Applying Theorem 1 and denoting the sequences $\{\mathbf{X}_1, \ldots, \mathbf{X}_t\}$ and $\{i_1, \ldots, i_t\}$ by $\mathbf{X}_1^t$ and i_i^t, respectively, the decision function that minimizes the probability of error is

$$\begin{aligned} g^*(\mathbf{x}_1^t) &= \arg\max_{i_1^t} \log P_{Y_1^t|\mathbf{X}_1^t}(i_1^t|\mathbf{x}_1^t) \\ &= \arg\max_{i_1^t} \{\log P_{\mathbf{X}_1^t|Y_1^t}(\mathbf{x}_1^t|i_1^t) + \log P_{Y_1^t}(i_1^t)\}, \end{aligned} \tag{7.1}$$

where the maximum is taken over all the possible configurations of Y_1^t. This is a well known problem in many areas of engineering and statistics including dynamics systems [54], speech processing [140], statistical learning [97], information theory [52] and, more recently, machine vision [28, 122, 196], where Y is a variable that encodes the state of the world and $\mathbf{x}$ observations from a phenomena to be modeled.

Application of the chain rule of probability leads to

$$\begin{aligned} g^*(\mathbf{x}_1^t) = \arg\max_{i_1^t} \{ & \log P_{\mathbf{X}_1|Y_1}(\mathbf{x}_1|i_1^t) + \log P_{Y_1}(i_1) \\ & + \sum_{k=2}^{t} [\log P_{\mathbf{X}_k|\mathbf{X}_1^{k-1}, Y_1^t}(\mathbf{x}_k|\mathbf{x}_1^{k-1}, i_1^t) + \log P_{Y_k|Y_1^{k-1}}(i_k|i_1^{k-1})]\}. \end{aligned}$$

Since, in practice, it is difficult to estimate the conditional probabilities $P_{\mathbf{X}_k|\mathbf{X}_1^{k-1}, Y_1^t}(\mathbf{x}_k|\mathbf{x}_1^{k-1}, i_1^t)$ and $P_{Y_k|Y_1^{k-1}}(i_k|i_1^{k-1})$ for large t (due to the combinatorial explosion of the number of possibilities for the conditioning event), it is usually necessary to rely on simplifying assumptions. A common solution is to rely on the following conditional independence assumption for the observations.

[1]Notice that, as in previous chapters, each query $\mathbf{x}_i$ is a collection of N_i feature vectors that we now denote by $\mathbf{x}_i = \{\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,N_i}\}$.

Assumption 5 *Given the target image class for iteration k, the query for that iteration is independent of the queries and target image classes for all other iterations*

$$P_{\mathbf{X}_k|\mathbf{X}_1^{k-1},Y_1^t}(\mathbf{x}_k|\mathbf{x}_1^{k-1}, i_1^t) = P_{\mathbf{X}_k|Y_k}(\mathbf{x}_k|i_k).$$

In the retrieval context, the assumption of conditional independence implies that the user provides the retrieval system with new information at each iteration. This is reasonable since users will tend to get frustrated if they feel that they have to repeat themselves, and will probably stop using the system when conditional independence does not hold.

Under Assumption 5, we obtain what is usually referred to as an *hidden Markov model* (HMM) [140] or a *Markov source* [52], and the probability of retrieval error is minimized when

$$g^*(\mathbf{x}_1^t) = \arg\max_{i_1^t}\{\log P_{Y_1}(i_1) + \sum_{k=1}^{t} \log P_{\mathbf{X}_k|Y_k}(\mathbf{x}_k|i_k) + \sum_{k=2}^{t} \log P_{Y_k|Y_1^{k-1}}(i_k|i_1^{k-1})\}.$$

This model is valid for both target search and open-ended browsing. When the transition probabilities $P_{Y_k|Y_1^{k-1}}(i_k|i_1^{k-1})$ are unconstrained, users are free to change their mind as frequently as they want and we have a model for browsing. If, on the other hand, switching between states is not allowed,

$$P_{Y_k|Y_1^{k-1}}(i_k|i_1^{k-1}) = P_{Y_k|Y_{k-1}}(i_k|i_{k-1}) = \delta_{i_{k-1},i_k},$$

where δ_{i_{k-1},i_k} is the Kronecker delta function defined by (2.3), we have a model for target search; i.e., the user decides on a target image class at the start of the interaction according to $P_{Y_1}(i_1)$ and holds on to that target class until it is found, or the search aborted. In this case, the retrieval model can be simplified into

$$g^*(\mathbf{x}_1^t) = \arg\max_{i}\{\log P_Y(i) + \sum_{k=1}^{t} \log P_{\mathbf{X}_k|Y}(\mathbf{x}_k|i)\}, \tag{7.2}$$

where $Y = Y_1$. In practice, estimating transition probabilities involves implementing an actual retrieval system, assembling a body of users and collecting extensive statistics on the patterns of interaction. This a "chicken and egg" problem since without the transition probabilities it is not possible to implement a system that supports browsing. For this reason, we restrict ourselves to the problem of target search, leaving the more general question of open-ended browsing open for subsequent discussion.

7.3 Target search

Using Assumption 5 the chain rule of probability and Bayes rule, (7.2) can also be written as

$$
\begin{aligned}
g^*(\mathbf{x}_1^t) &= \arg\max_i \{\log P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i) + \sum_{k=1}^{t-1} \log P_{\mathbf{X}_k|Y}(\mathbf{x}_k|i) + \log P_Y(i)\} \\
&= \arg\max_i \{\log P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i) + \sum_{k=1}^{t-1} \log P_{\mathbf{X}_k|\mathbf{X}_1^{k-1},Y}(\mathbf{x}_k|\mathbf{x}_1^{k-1}, i) + \log P_Y(i)\} \\
&= \arg\max_i \{\log P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i) + \log P_{\mathbf{X}_1^{t-1}|Y}(\mathbf{x}_1^{t-1}|i) + \log P_Y(i)\} \\
&= \arg\max_i \{\log P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i) + \log P_{Y|\mathbf{X}_1^{t-1}}(i|\mathbf{x}_1^{t-1})\}. \qquad (7.3)
\end{aligned}
$$

By comparing (7.3) with (2.14), the term $P_{Y|\mathbf{X}_1^{t-1}}(i|\mathbf{x}_1^{t-1})$ can be seen simply as a prior belief on the ability of the i^{th} image class to explain the query. However, unlike the straightforward application of the Bayesian criteria, this is not a static prior determined by some arbitrarily selected prior density. Instead, it is learned from the previous interaction between user and retrieval system and summarizes all the information in this interaction that is relevant for the decisions to be made in the future.

Recalling from (7.1) that, in target search mode,

$$g^*(\mathbf{x}_1^t) = \arg\max_i \{\log P_{Y|\mathbf{X}_1^t}(i|\mathbf{x}_1^t)\} \qquad (7.4)$$

and comparing with (7.3) reveals an intuitive mechanism to integrate information over time. Together, (7.3) and (7.4) state that the system's beliefs on the user's interests at time $t-1$ simply become the prior beliefs for iteration t. New data provided by the user at time t are then used to update these beliefs, generating the posteriors on which the retrieval decisions are based. These posteriors in turn become the priors for iteration $t+1$. In other words, prior beliefs are continuously updated from the observation of the interaction between user and retrieval system. This is illustrated in Figure 7.1.

From a computational standpoint, the procedure is very efficient since the bulk of the computation at each time step is due to the evaluation of the log-likelihood of the data in the corresponding query. Notice that this is exactly equation (2.16) and would have to be computed even in the absence of any learning. From the storage point of view, the efficiency is even higher since the entire interaction history is reduced to a number per image class. It

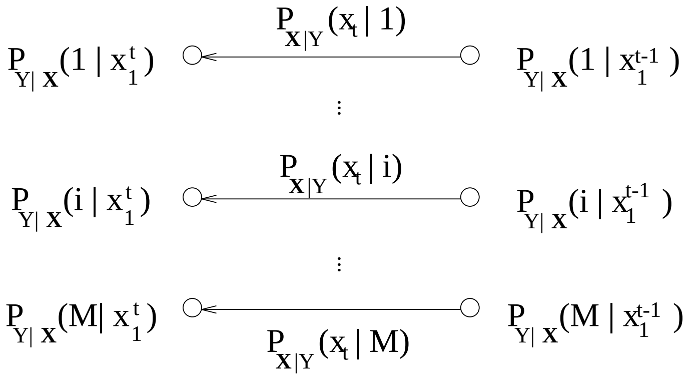

Figure 7.1: Belief propagation across iterations of the retrieval process.

is a remarkable fact that this number alone enables decisions that are optimal with respect to the entire interaction.

There is however one limitation associated with this belief propagation which is evident from (7.2): for large t, the contribution of the new data provided by the user is very small and the posterior probabilities tend to remain constant. This limitation can be avoided by replacing (7.3) with the more generic maximization problem [11],

$$g^*(\mathbf{x}) = \arg\max_i \{\log P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i) + \lambda \log P_{Y|\mathbf{X}_1^{t-1}}(i|\mathbf{x}_1^{t-1})\},$$

where one looks for the image class that best explains the current query, under a constraint on how well it explains all the previous interaction. The scalar λ is a Lagrange multiplier that weighs the importance of the past. Defining $\alpha = 1/(1+\lambda) \in [0,1]$ this is equivalent to

$$g^*(\mathbf{x}) = \arg\max_i \{\alpha \log P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i) + (1-\alpha) \log P_{Y|\mathbf{X}_1^{t-1}}(i|\mathbf{x}_1^{t-1})\}, \tag{7.5}$$

in which case the past is all that matters when $\alpha = 0$, while all the emphasis is on the current query when $\alpha = 1$, and we recover (7.3) when $\alpha = 0.5$. Rewriting this equation as

$$\begin{aligned} g^*(\mathbf{x}) &= \arg\max_i \{\alpha \log P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i) + \alpha(1-\alpha) \log P_{\mathbf{X}_{t-1}|Y}(\mathbf{x}_{t-1}|i) \\ &\quad + (1-\alpha)^2 \log P_{Y|\mathbf{X}_1^{t-2}}(i|\mathbf{x}_1^{t-2})\} \\ &= \arg\max_i \{\sum_{k=1}^{t} \alpha(1-\alpha)^{t-k} \log P_{\mathbf{X}_k|Y}(\mathbf{x}_k|i) + (1-\alpha)^t \log P_Y(i)\} \end{aligned}$$

$\alpha(1-\alpha)^{t-k}$ can be seen as a *decay factor* that penalizes older terms.

7.4 Negative feedback

While positive feedback is a powerful addition to retrieval systems, there are many situations in CBIR where it is not sufficient to guarantee satisfactory performance. In such situations, it is usually possible to attain significant improvements by combining it with negative feedback. One example is when various image classes in the database have overlapping densities. This is illustrated in Figure 7.2, where we depict an hypothetical search on a database with two major image classes that share a common attribute (large regions of blue sky), but are different in other aspects (images in class A also contain regions of white snow, while those in class B contain regions of grass). This could, for example, be a database of recreation sites where class A contains pictures of a ski resort, while class B contains pictures of the same resort but taken during the summer.

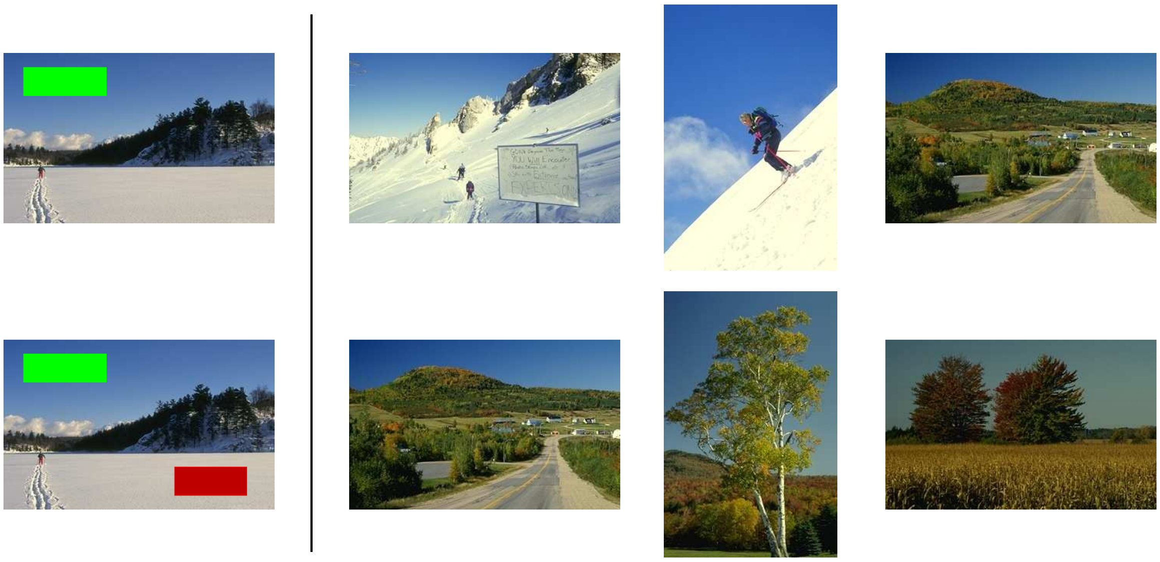

Figure 7.2: Two queries based on the same query image (shown on the left). The regions blocked by the light green (dark red) rectangle are positive (negative) examples for the search. Top: query for sky. Bottom: query for sky but not snow.

If the user starts with an image of class A (e.g. a picture of a snowy mountain), using regions of sky as positive examples is not likely to quickly lead to the images of class B. In fact, all other factors being equal, there is an equal likelihood that the retrieval system will return images from the two classes. This is illustrated in the top row. On the other hand, if the user can explicitly indicate interest in regions of sky but not in regions of snow, the likelihood that only images from class B will be returned increases drastically. This is illustrated in the bottom row.

Another example of the importance of negative feedback are situations in which there does not appear to be a good positive example to select next. These happen when, in response to user feedback, the system returns a collection of images that have already been retrieved in previous iterations. Assuming the user has already given the system all the possible positive feedback, the only way to escape from such situations is to choose some regions that are not desirable and use them as negative feedback. In the example above, when users get stuck with a screen full of pictures of white mountains, they can simply select some regions of snow to escape the local minima. On the other hand, if only positive examples were allowed, what to do next would not be clear. This is illustrated in Figure 7.3.

Figure 7.3: The user is looking for pictures taken in the summer and has already provided the retrieval system with various examples of sky. What to do next is not clear, unless negative examples are allowed.

In order to account for negative examples, we must penalize the classes under which these score well while favoring the classes that assign a high score to the positive examples. Unlike positive examples, for which the likelihood is known, it is not straightforward to estimate the likelihood of a particular negative example given that the user is searching for a certain image class. We denote the use of the vector $\mathbf{z}$ as a negative example by $\bar{\mathbf{z}}$ and rely on the following assumption.

Assumption 6 *The likelihood that* $\mathbf{z}$ *will be used as a negative example, given that the target is class* i*, is equal to the likelihood it will be used as a positive example given that the target is any other class*

$$P_{\mathbf{Z}|Y}(\bar{\mathbf{z}}|Y=i) = P_{\mathbf{Z}|Y}(\mathbf{z}|Y \neq i). \tag{7.6}$$

This assumption captures the intuition that, when searching for class i, a good negative example is one that would be a good positive example if the user were looking for any class other than i. For example, if class i is the only one that does not contain regions of sky, using pieces of sky as negative examples will quickly eliminate the other images in the database. Thus, one would expect the user to provide sky as a negative example with high probability.

Denoting by $\bar{\mathbf{z}}_1^t = \{\bar{\mathbf{z}}_i\}_{i=1}^t$ the collection of negative queries, these can be accounted for by simply replacing (7.3) and (7.4) with

$$\begin{aligned}
g^*(\mathbf{x}_1^t) &= \arg\max_i \{\log P_{Y|\mathbf{X},\mathbf{Z}}(i|\mathbf{x}_1^t, \bar{\mathbf{z}}_1^t)\} \\
&= \arg\max_i \{\log P_{\mathbf{X}_t,\mathbf{Z}_t|Y}(\mathbf{x}_t, \bar{\mathbf{z}}_t|i) + \log P_{Y|\mathbf{X}_1^{t-1},\mathbf{Z}_1^{t-1}}(i|\mathbf{x}_1^{t-1}, \bar{\mathbf{z}}_1^{t-1})\} \\
&= \arg\max_i \{\log P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i) + \log P_{\mathbf{Z}_t|Y}(\bar{\mathbf{z}}_t|i) + \log P_{Y|\mathbf{X}_1^{t-1},\mathbf{Z}_1^{t-1}}(i|\mathbf{x}_1^{t-1}, \bar{\mathbf{z}}_1^{t-1})\} \\
&= \arg\max_i \{\log P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i) + \log P_{\mathbf{Z}_t|Y}(\mathbf{z}_t|Y \neq i) + \log P_{Y|\mathbf{X}_1^{t-1},\mathbf{Z}_1^{t-1}}(i|\mathbf{x}_1^{t-1}, \bar{\mathbf{z}}_1^{t-1})\} \\
&= \arg\max_i \left\{\log P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i) + \log \frac{P_{Y|\mathbf{Z}_t}(Y \neq i|\mathbf{z}_t)}{P_Y(Y \neq i)} + \log P_{Y|\mathbf{X}_1^{t-1},\mathbf{Z}_1^{t-1}}(i|\mathbf{x}_1^{t-1}, \bar{\mathbf{z}}_1^{t-1})\right\} \\
&= \arg\max_i \left\{\log P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i) + \log \frac{1 - P_{Y|\mathbf{Z}_t}(i|\mathbf{z}_t)}{1 - P_Y(i)} + \log P_{Y|\mathbf{X}_1^{t-1},\mathbf{Z}_1^{t-1}}(i|\mathbf{x}_1^{t-1}, \bar{\mathbf{z}}_1^{t-1})\right\}
\end{aligned}$$

where we have also used Assumptions 5 and 6 and the fact that $P_{\mathbf{Z}_t}(\mathbf{z}_t)$ does not depend on i. Applying Bayes rule recursively then leads to the following natural generalization of (7.2)

$$g^*(\mathbf{x}_1^t) = \arg\max_i \left\{\log P_Y(i) + \sum_{k=1}^{t} \log P_{\mathbf{X}_k|Y}(\mathbf{x}_k|i) + \log \frac{1 - P_{Y|\mathbf{Z}_k}(i|\mathbf{z}_k)}{1 - P_Y(i)}\right\}.$$

In practice, however, this equation is not very useful since the terms $1 - P_{Y|\mathbf{Z}_k}(i|\mathbf{z}_k)$ and $1 - P_Y(i)$ tend to be close to one. To see this, suppose that we are dealing with a database of $10,000$ image classes which are assumed to be equally likely a priori: $P_Y(i) = 1/10,000$. In this case, even if the observation of $\mathbf{z}_k$ increases, the probability of class i one-hundred-fold

$P_{Y|\mathbf{Z}_k}(i|\mathbf{z}_k) = 1/100$, the ratio $(1 - P_{Y|\mathbf{Z}_k}(i|\mathbf{z}_k))/(1 - P_Y(i))$ is only 0.99. This means that negative examples have very small influence in the overall decision function.

An alternative solution is to choose the class i that maximizes the *posterior odds ratio* [55] between the hypotheses "class i is the target" and "class i is not the target"

$$\begin{aligned}
g^*(\mathbf{x}_1^t) &= \arg\max_i \log \frac{P_{Y|\mathbf{X}_1^t,\mathbf{Z}_1^t}(i|\mathbf{x}_1^t, \bar{\mathbf{z}}_1^t)}{P_{Y|\mathbf{X}_1^t,\mathbf{Z}_1^t}(Y \neq i|\mathbf{x}_1^t, \bar{\mathbf{z}}_1^t)} \\
&= \arg\max_i \log \left(\frac{P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i)}{P_{\mathbf{X}_t|Y}(\mathbf{x}_t|Y \neq i)} \frac{P_{\mathbf{Z}_t|Y}(\bar{\mathbf{z}}_t|i)}{P_{\mathbf{Z}_t|Y}(\bar{\mathbf{z}}_t|Y \neq i)} \frac{P_{Y|\mathbf{X}_1^{t-1},\mathbf{Z}_1^{t-1}}(i|\mathbf{x}_1^{t-1}, \bar{\mathbf{z}}_1^{t-1})}{P_{Y|\mathbf{X}_1^{t-1},\mathbf{Z}_1^{t-1}} P(Y \neq i|\mathbf{x}_1^{t-1}, \bar{\mathbf{z}}_1^{t-1})} \right) \\
&= \arg\max_i \log \left(\frac{P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i)}{P_{\mathbf{X}_t|Y}(\mathbf{x}_t|Y \neq i)} \frac{P_{\mathbf{Z}_t|Y}(\mathbf{z}_t|Y \neq i)}{P_{\mathbf{Z}_t|Y}(\mathbf{z}_t|i)} \frac{P_{Y|\mathbf{X}_1^{t-1},\mathbf{Z}_1^{t-1}}(i|\mathbf{x}_1^{t-1}, \bar{\mathbf{z}}_1^{t-1})}{P_{Y|\mathbf{X}_1^{t-1},\mathbf{Z}_1^{t-1}}(Y \neq i|\mathbf{x}_1^{t-1}, \bar{\mathbf{z}}_1^{t-1})} \right) \\
&= \arg\max_i \log \left(\frac{P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i)}{P_{\mathbf{Z}_t|Y}(\mathbf{z}_t|i)} \frac{P_{Y|\mathbf{Z}_t}(Y \neq i|\mathbf{z}_t)}{P_{Y|\mathbf{X}_t}(Y \neq i|\mathbf{x}_t)} \frac{P_{Y|\mathbf{X}_1^{t-1},\mathbf{Z}_1^{t-1}}(i|\mathbf{x}_1^{t-1}, \bar{\mathbf{z}}_1^{t-1})}{P_{Y|\mathbf{X}_1^{t-1},\mathbf{Z}_1^{t-1}}(Y \neq i|\mathbf{x}_1^{t-1}, \bar{\mathbf{z}}_1^{t-1})} \right) \\
&= \arg\max_i \log \left(\frac{P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i)}{P_{\mathbf{Z}_t|Y}(\mathbf{z}_t|i)} \left[\frac{1 - P_{Y|\mathbf{Z}_t}(i|\mathbf{z}_t)}{1 - P_{Y|\mathbf{X}_t}(i|\mathbf{x}_t)} \right] \frac{P_{Y|\mathbf{X}_1^{t-1},\mathbf{Z}_1^{t-1}}(i|\mathbf{x}_1^{t-1}, \bar{\mathbf{z}}_1^{t-1})}{P_{Y|\mathbf{X}_1^{t-1},\mathbf{Z}_1^{t-1}}(Y \neq i|\mathbf{x}_1^{t-1}, \bar{\mathbf{z}}_1^{t-1})} \right) \\
&\approx \arg\max_i \left\{ \log \frac{P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i)}{P_{\mathbf{Z}_t|Y}(\mathbf{z}_t|i)} + \log \frac{P_{Y|\mathbf{X}_1^{t-1},\mathbf{Z}_1^{t-1}}(i|\mathbf{x}_1^{t-1}, \bar{\mathbf{z}}_1^{t-1})}{P_{Y|\mathbf{X}_1^{t-1},\mathbf{Z}_1^{t-1}}(Y \neq i|\mathbf{x}_1^{t-1}, \bar{\mathbf{z}}_1^{t-1})} \right\}
\end{aligned}$$

where we have used the fact that $(1 - P_{Y|\mathbf{Z}_t}(i|\mathbf{z}_t))/(1 - P_{Y|\mathbf{X}_t}(i|\mathbf{x}_t)) \approx 1$. Including a decay factor to penalize ancient terms, we obtain

$$g^*(\mathbf{x}_1^t) \approx \arg\max_i \left\{ \alpha \log \frac{P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i)}{P_{\mathbf{Z}_t|Y}(\mathbf{z}_t|i)} + (1-\alpha) \log \frac{P_{Y|\mathbf{X}_1^{t-1},\mathbf{Z}_1^{t-1}}(i|\mathbf{x}_1^{t-1}, \bar{\mathbf{z}}_1^{t-1})}{P_{Y|\mathbf{X}_1^{t-1},\mathbf{Z}_1^{t-1}}(Y \neq i|\mathbf{x}_1^{t-1}, \bar{\mathbf{z}}_1^{t-1})} \right\}. \quad (7.7)$$

This equation is similar to (7.5) but now the terms on the denominator penalize the image classes that explain well the negative examples. Overall, the decision function favors image classes that explain well the positive examples and poorly the negative ones.

There is however, under the posterior odds ratio, a tendency to over-emphasize the importance of negative examples. In particular, any class with zero probability of generating the negative examples will lead to an infinite ratio, even if it explains very poorly the positive examples. To avoid this problem, we proceed in two steps:

- start by solving (7.5), i.e. sort the classes according to how well they explain the positive examples.

- select the subset of the best N classes and solve (7.7) considering only the classes in this subset;

Overall, the inference algorithm has two free parameters: the decay factor α, and the size N of the window used in the second step. In the next section, we present experimental evidence that the learning performance is quite robust to variations in these parameters.

7.5 Experimental evaluation

We performed several experiments to evaluate the improvements in retrieval performance achievable with Bayesian inference. Because in an ordinary browsing scenario it is difficult to know the ground truth for the retrieval operation (at least without going through the tedious process of hand-labeling all images in the database), we relied on the mosaic databases (for which ground truth is available).

7.5.1 Experimental setup

The goal of the experiments was to determine if it is possible to reach a desired target image by starting from a weakly related one and providing feedback to the retrieval system. This simulates the interaction between a real user and the CBIR system and is an iterative process, where each iteration consists of 1) selecting a few examples, 2) using them as queries for retrieval, and 3) examining the top V retrieved images to find examples for the next iteration. V should be small since most users are not willing to go through lots of false positives to find the next query.

The most challenging problem in automated testing is to determine a good strategy for selecting the examples to be given to the system. The closer this strategy is to what a real user would do, the higher the practical significance of the results. However, even when there is clear ground truth for the retrieval (as is the case of the mosaic databases), it is not completely clear how to make the selection. While it is obvious that regions of texture or object classes that appear in the target should be used as positive feedback, it is much harder to determine automatically what are good negative examples. As shown in

Figure 7.4, there are cases in which images from two different classes are visually similar. Selecting images from one of these classes as a negative example for the other will be a disservice to the learner.

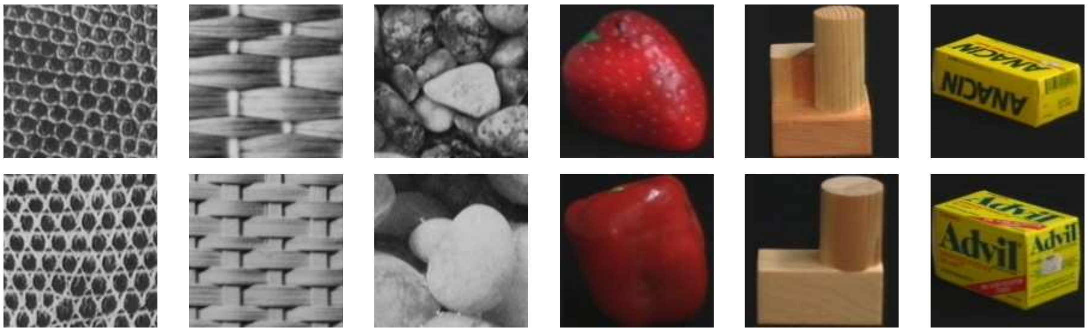

Figure 7.4: Examples of pairs of visually similar images that appear in different image classes.

While real users tend not to do this, it is hard to avoid such mistakes in an automated setting, unless one does some sort of pre-classification of the database. Because we wanted to avoid such pre-classification, we decided to stick with a simple selection procedure and live with these mistakes. At each step of the iteration, examples were selected in the following way: among the top V images returned by the retrieval system, the one with most sub-images from image classes also present in the target was selected to be the next query. One block from each sub-image in the query was then used as a positive (negative) example if the texture or object depicted in that sub-image was also (was not) represented in the target image.

This strategy is a worst-case scenario. First, the learner might be confused by conflicting negative examples. Second, as seen in Chapter 6, better retrieval performance can be achieved if more than one block from each region is included in the queries. However, using only one block reduces the computational complexity of each iteration, allowing us to 1) average results over several runs of the inference process and 2) experiment several values for the α and V parameters of the retrieval system. We selected several (α, V) pairs and performed 100 runs with random target images for each. In all cases, the initial query image was the first in the database containing one sub-image in common with the target.

The performance of the learning algorithm can be evaluated in various ways. We considered two metrics: the percentage of the runs that converged to the right target and the number of iterations required for convergence. Because, to prevent the learner from entering loops, any given image could only be used once as a query, the algorithm can diverge in two ways. Strong divergence occurs when, at a given time step, the images (among the top V) that can be used as queries do not contain any sub-image in common with the target. In such situation, a real user will tend to feel that the retrieval system is incoherent and abort the search. Weak divergence occurs when all the top V images have previously been used. This is a less troublesome situation because the user could simply look up more images (e.g. the next V) to get new examples. To make the presentation compact, in the next sections we only show results relative to strong divergence.

7.5.2 Positive feedback

Figure 7.5 presents plots of the convergence rate, mean number of iterations until convergence, and number of iterations until divergence as a function of the decay factor α and the number of matches V, for the two mosaic databases. In both cases, the inclusion of learning ($\alpha < 1$) typically increases the convergence rate. This increase can be very significant (as high as 15%), and larger gains occur when the convergence rate without learning is low. If the convergence rate is already high without learning, the inclusion of learning does not change it significantly. In general, a precise selection of α is not crucial for achieving a rate of convergence close to the best possible.

The rate of convergence is also affected by the number of matches V from which the user is allowed to select the next query. While, as expected, the larger this number the faster the convergence, a law of diminishing returns seems to be in effect: while the convergence rate increases quickly when V is small, it levels off for large values of V. This is an interesting result because there is usually a cost associated with a large V: users are not willing to go through lots of image screens in order to find a suitable next query. Another interesting result is that, under the assumption that users will only look at the first image screen returned by the retrieval system ($V \in [15, 20]$), the inclusion of learning leads to visible convergence improvements.

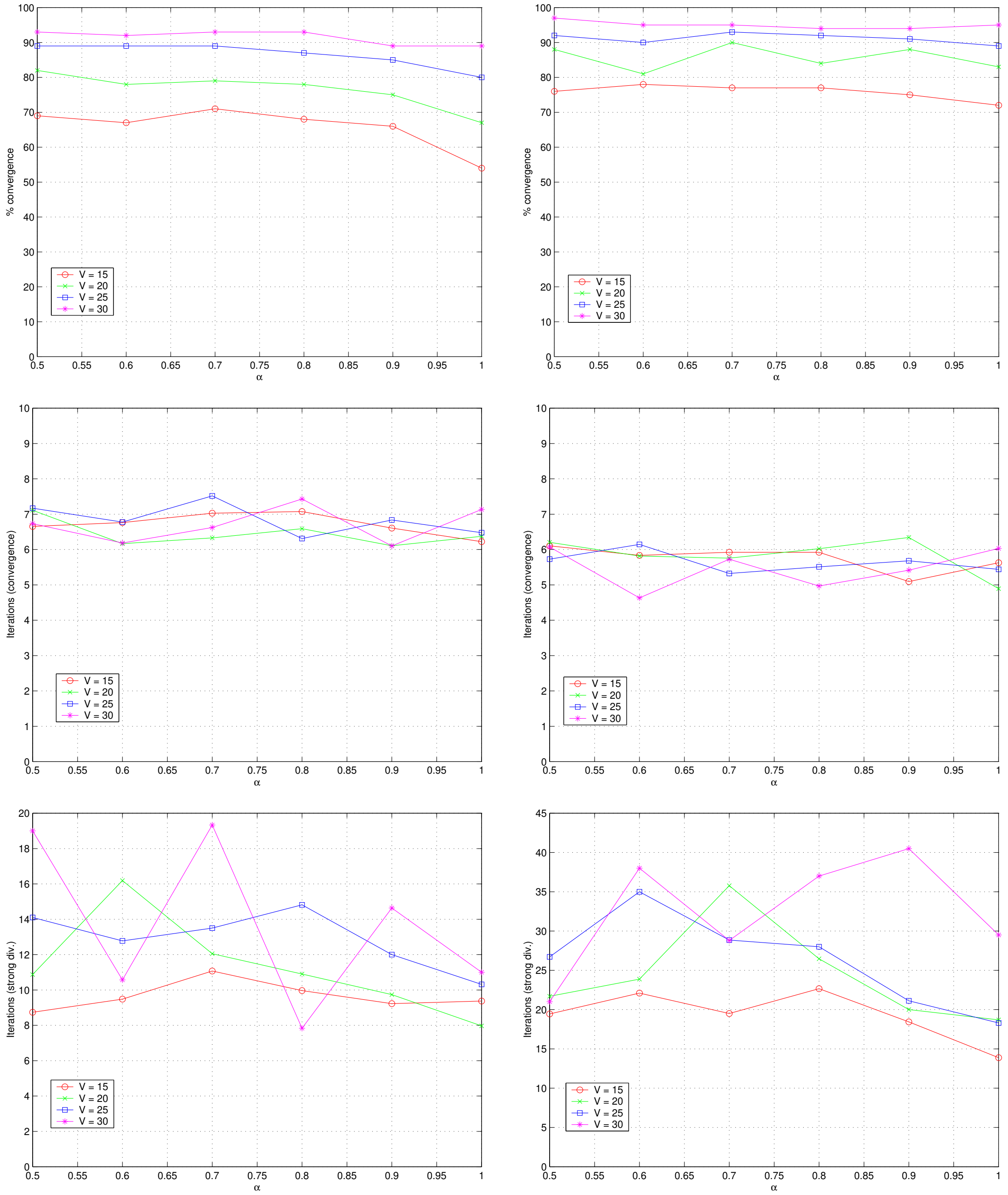

Figure 7.5: Plots, as a function of the learning factor α, of the convergence rate (top) average number of iterations for convergence (middle) and divergence (bottom). In all plots different curves correspond to different values for the number of images V examined at the end of each iteration. Left: Brodatz mosaic database. Right: Columbia mosaic database.

In terms of the number of iterations, when convergence occurs it is usually very fast (from 4 to 8 iterations). On the other hand, the number of iterations until divergence is usually well above 8. This indicates that, in practice, it would be easy to detect when the retrieval system is not likely to converge: if the number of iterations is above 8 to 10, then it is probably preferable to start a new query (from another initial image) than to insist on the current one.

We next present examples of the relevance feedback process in the Columbia mosaic database. Figure 7.6 depicts a search for a target image consisting of a plastic bottle, a white hanger, a blue plastic donut, and a coffee cup. The top picture depicts the first iteration of the retrieval process. The search starts with an image containing the blue donut and, since the retrieval precision is high for this object, it appears in all the 15 retrieved image slots. At this point the retrieval system has basically restricted the set of possible matches to the K images that contain the blue donut. Hence, for each of the 15 slots, the probability of the target image appearing in the slot is approximately $1/K$. Since $K << D$, where D is the database size, this is dramatically higher than the $1/D$ associated with random guessing. Furthermore, if there are M images containing both the blue donut and one of the three other objects of interest, the corresponding probability is M/K. This can be high and it is therefore not surprising that, due to chance alone, one of the other objects in the target will also appear among the retrieved images. This is indeed the case of the first image in the second row, which also contains the white hanger.

By the selecting this image as next query (bottom picture) the user increases the probability per slot from approximately $1/K$ to approximately $1/L$ where L is the total number of images containing the two query objects. Notice that, as more objects are included in the query, the total number of images containing those objects decreases substantially and the above probabilities increase drastically. In the example of the figure, because there are less than 15 images in the entire database containing both objects, one would expect the target to appear among the top 15 matches with very high probability. This is indeed the case, and the target shows up as the first image in the second row.

The example illustrates how, provided that precision is high for the individual query objects, retrieval can be very fast. In this sense, it corresponds to a best-case scenario since

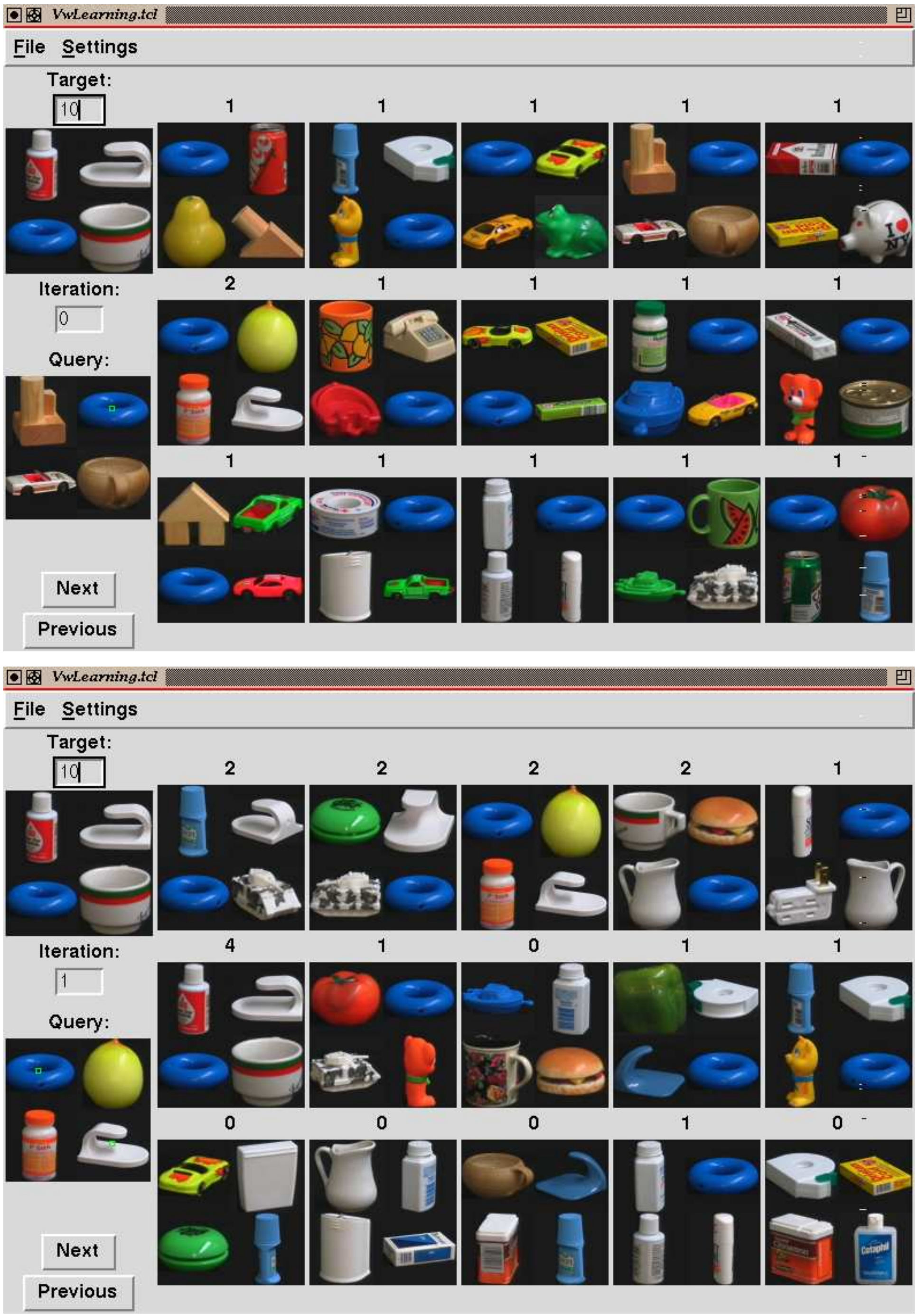

Figure 7.6: Two iterations of positive feedback. In both cases, the target image is shown at the top left and the query image immediately below. The query itself is based on a single feature vector from each of the sub-images (8×8 neighborhood indicated in the center of the sub-image) that are shared by the query and the target images. The number above each retrieved image indicates the number of objects that it shares with the target.

the system will typically have to deal with objects for which precision is not so high. It is in these situations that learning becomes most important.

Figures 7.7 and 7.8 show one such example. The target consists of a plastic bottle, a container of adhesive tape, a clay cup, and a white mug, while the initial query is an image containing the clay cup. Since there are various objects made of wood in the Columbia database and these have surface properties visually similar to those of the clay cup, precision is now significantly smaller (top picture of Figure 7.7): only 4 of the 15 top matches are correct. This makes it difficult to zero in on the target for two fundamental reasons: first, it is not as likely as in the previous example that the other objects in the target will appear among the top matches. Second, when this happens, it is likely that the new target objects will not share an image with the object used in the query. Both of these points are illustrated by the example. First, the feedback process must be carried for three iterations before a target object other than that in the query appears among the top matches. Second, when this happens (top picture of Figure 7.8), the new object (tape container) is not part of an image that also contains the clay cup.

In this situation, the most sensible option is to base the new query on the newly found target object (tape container). However, in the absence of learning, it is unlikely that the resulting matches will contain any instances of the query object used on the previous iterations (clay cup) or the objects that are confounded with it. As illustrated by the bottom picture of Figure 7.8, the role of learning is to favor images containing these objects. In the example of the figure, 7 of the 15 images returned in response to a query based on the tape container include the clay cup or visually similar objects (in addition to the tape container itself). This enables new queries based on both target objects which, as seen in the previous example, have an increased chance of success. In this particular case, it turns out that one of the returned images is the target itself.

7.5.3 Negative feedback

Figure 7.9 presents plots of the convergence rate, mean number of iterations until convergence, and strong divergence rate for the two mosaic databases when both positive and negative feedback are used and the number, N, of top positive feedback matches considered

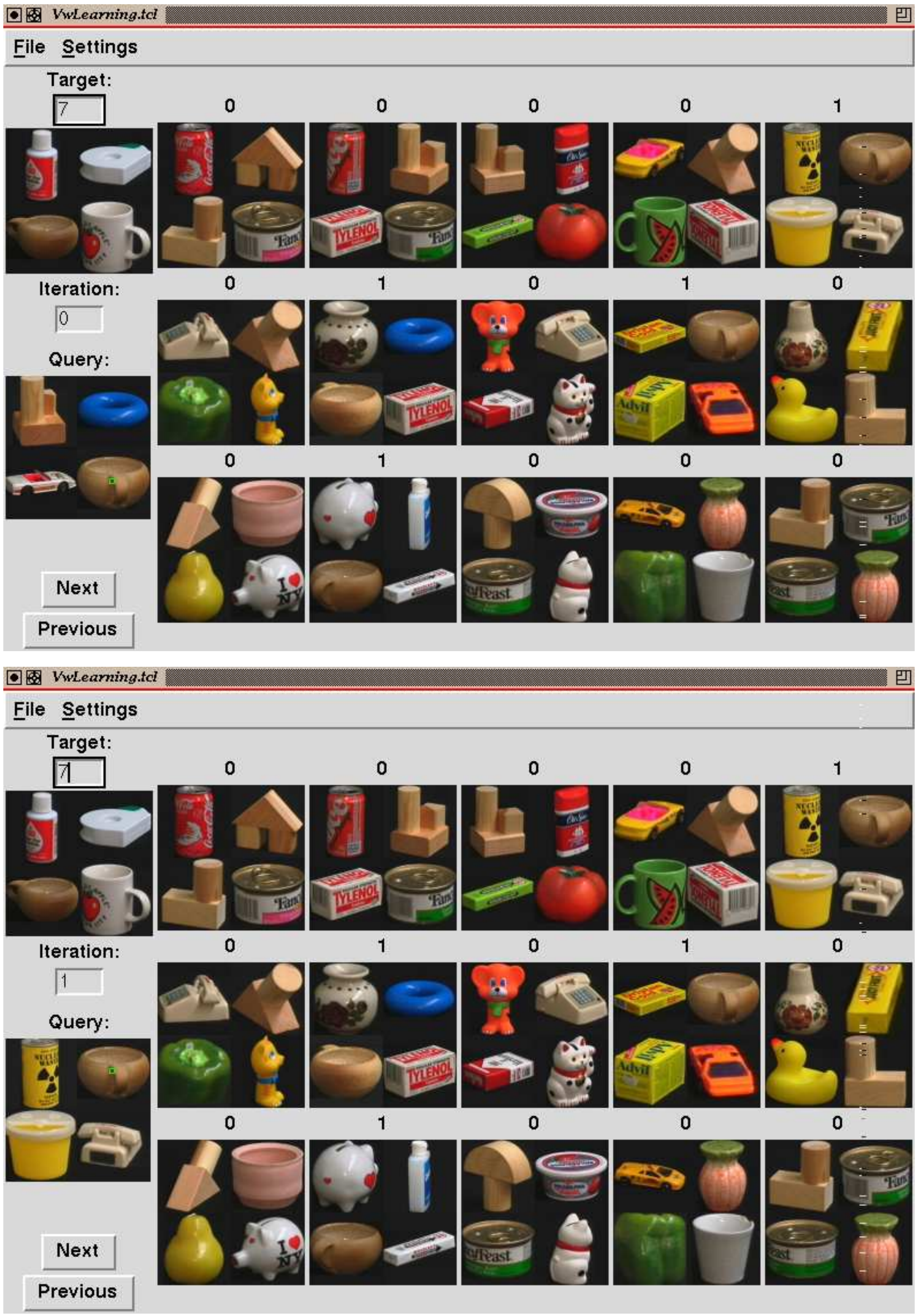

Figure 7.7: First two iterations of positive feedback for the example discussed in the text.

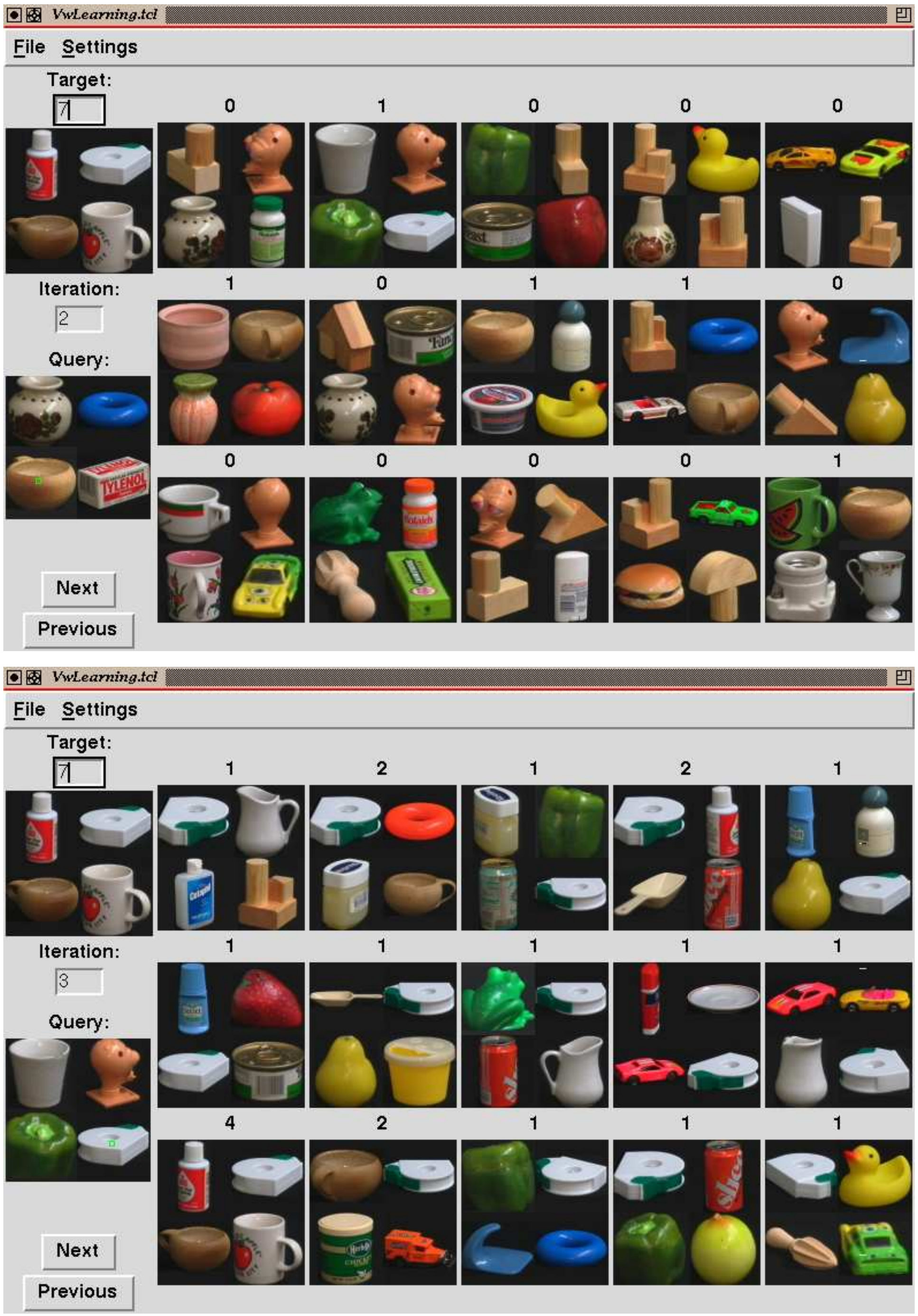

Figure 7.8: Last two iterations of positive feedback for the example discussed in the text.

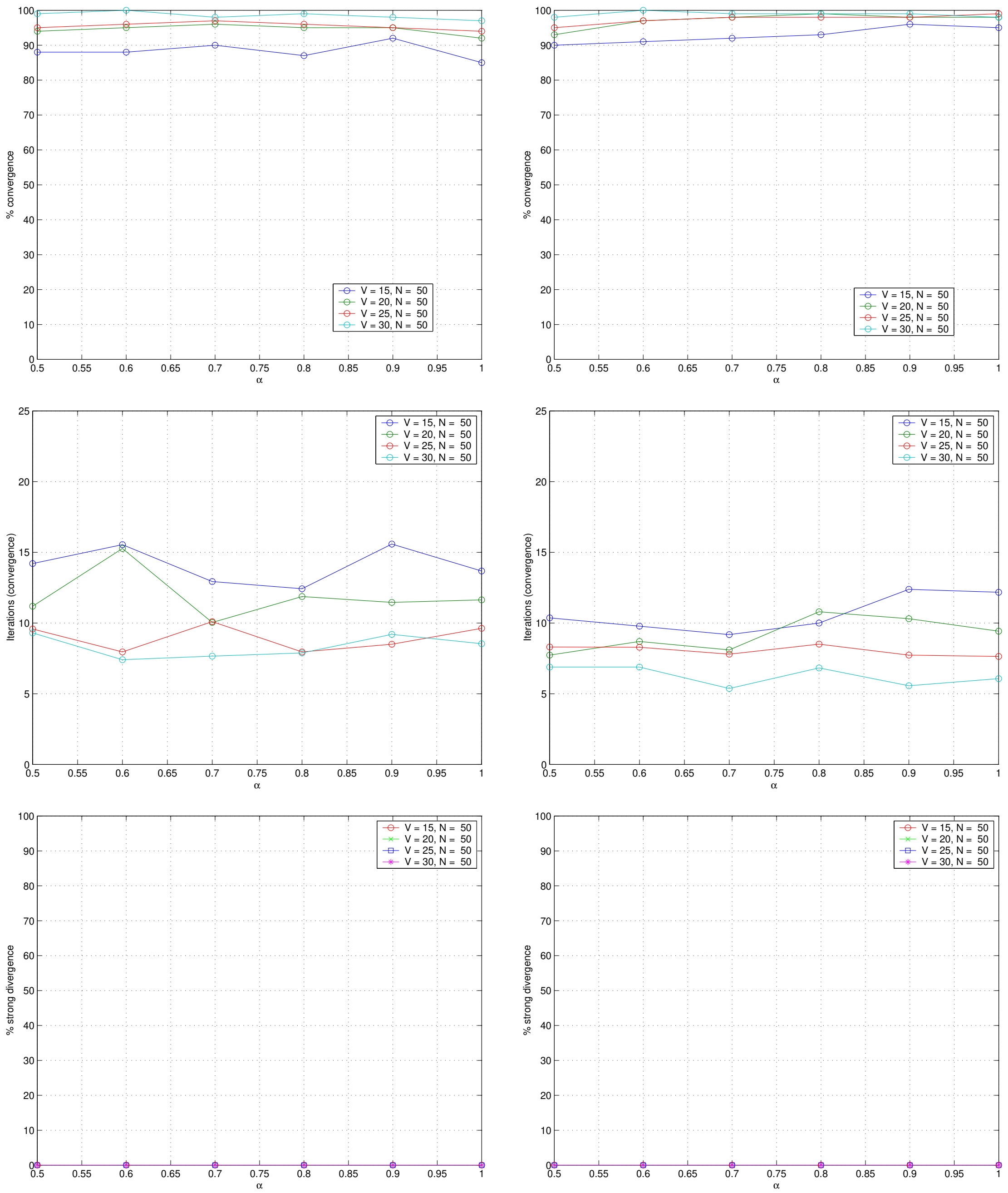

Figure 7.9: Plots, as a function of the learning factor α, of the convergence rate (top), average number of iterations for convergence (middle), and divergence rate (bottom). In all plots, different curves correspond to different values of the number of images V examined at the end of each iteration and N is number of top positive feedback matches that are considered when negative feedback is taken into account. Left: Brodatz mosaic database. Right: Columbia mosaic database.

in (7.7) is 50. Comparing with the plots of Figure 7.5, it is clear that the convergence rate is significantly improved by the inclusion of negative feedback. In particular, for most values of V the convergence rate is close to 100% and, in all cases, the rate of strong divergence is zero.

Since the convergence rate is high, learning is usually less relevant than it was in the experiments where only positive feedback is allowed. Notice, however, that the best convergence happens for values of α larger than 0.5. In fact, some degradation is noticeable on Columbia as we approach the 0.5 limit. This degradation is related to the fact that the number of iterations required for convergence is now larger than for positive feedback-only retrieval. This increase in the number of iterations is more significant for the harder retrieval scenarios (smaller V) and particularly noticeable for $V = 15$, where it reaches 100% on Brodatz and and 50% on Columbia.

The combination of all these observations allows the following conclusions. First, introducing negative feedback allows *exploration* of the database and this, in turn, leads to better convergence. In particular, retrieval is never stopped because the user runs out of examples to select next. Second, an increase in the number of iterations required for convergence is inherent to this exploration. Notice that, while in Figure 7.5 this number was approximately the same for all V, we now have significant differences. In particular, convergence takes longer for smaller V, i.e. when retrieval is most difficult. This is a sensible result, and suggests that the average number of iterations increases because retrieval takes much longer on a small set of difficult cases. Our personal experience of interaction with the retrieval system confirms this hypothesis.

The plots also expose a trade-off between the number of images that the user inspects to find the next query (V) and the number of iterations for convergence. Notice that, by increasing V from 15 (approximately one screen of images) to 30 (two screens), it is possible to increase the speed of convergence to the levels of Figure 7.5. It remains to be studied what would be more appealing to users: less iterations or a smaller number of images to inspect per iteration.

Figure 7.10 shows the impact of the parameter N in the retrieval performance. The figure depicts the convergence rate and speed for the Columbia mosaic database when

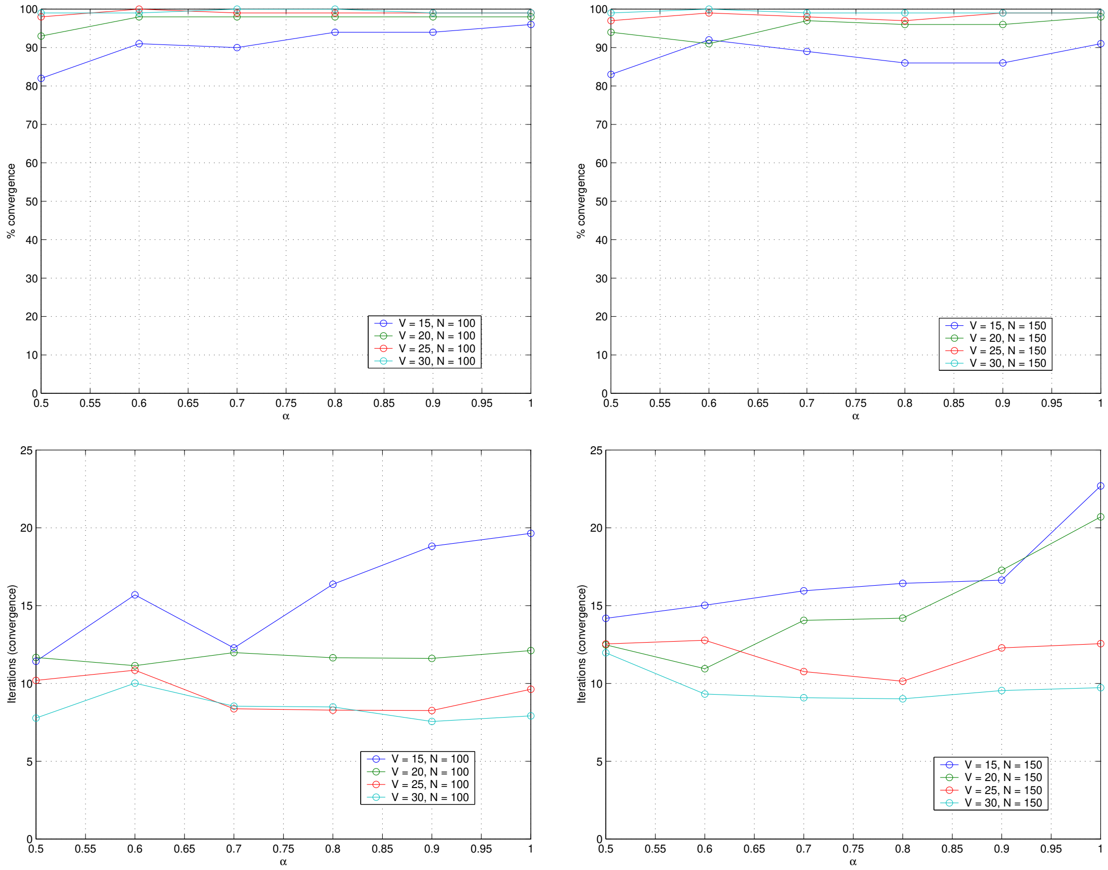

Figure 7.10: Plots, as a function of the learning factor α and number of images V examined at the end of each iteration, of the convergence rate (top) and average number of iterations for convergence (bottom) on the Columbia mosaic database. N is number of top positive feedback matches that are considered when negative feedback is taken into account. Left: $N = 100$. Right: $N = 150$.

$N = 100$ and $N = 150$. Notice that, while performance is not as good as when $N = 50$, it is still clearly superior to that achievable with positive feedback alone. This confirms that a very precise selection of N is not required to guarantee the improvements inherent to negative feedback. More drastic differences happen for the convergence speed, which can vary substantially with N. Here, however, learning plays a significant role and, when learning is in effect, the number of iterations necessary for convergence can also be reduced to the levels of Figure 7.5.

We finish by presenting some examples of how negative feedback can indeed improve the speed of convergence to the target image. Figure 7.11 depicts a search for a target image consisting of a rubber frog, a toy boat, a plastic jar, and a plastic bottle. The top picture depicts the first iteration of the retrieval process when only positive feedback is allowed. The bottom picture depicts the same iteration when both positive and negative feedback are used. The search starts with an image containing two views of the rubber frog, a plastic donut, and a clay object.

Observation of the top picture reveals that there are several images in the database where the rubber frog appears along with wooden objects, that have surface properties similar to those of clay. When negative feedback is allowed and the clay object used as a negative example, these images are penalized and the rank of the target image improves significantly. Notice that, while for positive feedback only (top image) seven slots are occupied by objects that have similar surface properties to those of the clay object, only two appear when negative examples are also allowed (bottom image). Consequently, the rank of the target image improves from higher than 15 to 6.

This example illustrates the importance of negative examples when dealing with overlapping images classes, as we had already suggested through Figure 7.2. For the mosaic databases, overlap means images sharing the same objects. In this particular query, convergence takes 6 iterations when the retrieval system is based on positive feedback alone and 1 iteration when negative examples are also allowed.

The final example (Figures 7.12 and 7.13) illustrates the importance of negative examples when it is not clear what positive examples to choose next. The target image is now composed of a blue hanger, a stapler, a clay object, and a plastic jar. The search starts

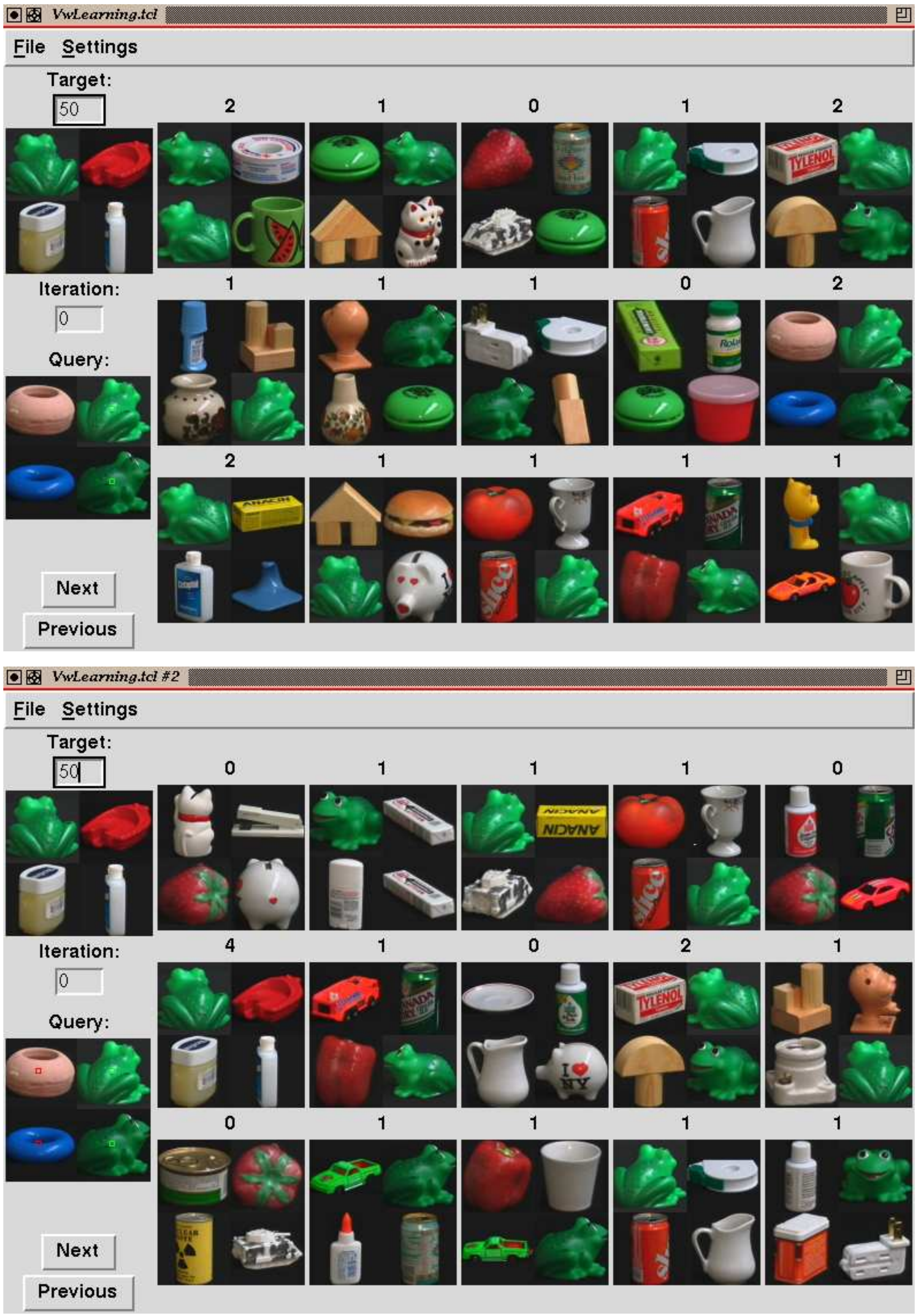

Figure 7.11: First iteration of relevance feedback for the same query image when only positive (top) and both positive and negative feedback (bottom) are allowed. The query itself is based on a single feature vector from each of the sub-images (8×8 neighborhood indicated in the center of the sub-image). Positive examples are extracted from the sub-images that are shared by the query and the target images, negative examples are extracted from the remaining sub-images.

with the query image that was also used to initiate the previous example (two rubber frogs, clay object, and plastic donut).

Figure 7.12 presents the first two iterations for the situation in which only positive feedback is allowed. The clay object is selected in the first iteration, and 15 images containing clay objects are returned. While this is an impressive result in terms of precision/recall, it is not very useful from the point of view of getting to the target image. In fact, it is not clear that any of the returned images is closer to the target than the query image itself. Since only positive examples are allowed, the only alternative is to choose another image containing the clay object (preferably under a different view than the previously used). This is exactly what happens, and an image containing two different views of the clay object is selected for the next query. However, since the new examples are not all that different from the ones used in the first iteration, the retrieved images are approximately the same. In fact, out of the 15 images returned in the second iteration only 4 had not been already retrieved in the first. This makes it even more difficult to decide on which image to use as next query. It appears that the retrieval system is not doing much progress and, in a real retrieval scenario, the user would tend to get frustrated. In fact, proceeding in this way takes 8 iterations to get to the target.

Figure 7.13 presents results for the same query when negative feedback is allowed. In this case, in addition to the clay objects, several ceramic objects are also returned in response to the first query. Since there are no ceramic objects in the target, such objects are a good selection to use as negative examples in the next iteration. This is what happens and, despite the fact that the query images used in the two iterations are the same as in Figure 7.12, the number of images retrieved in both iterations is now only 3. Since, as before, the positive examples constrain these images to contain clay objects, it is not surprising that the target is reached in the second iteration. This example confirms what had already been pointed out in Figure 7.3: negative examples allow users to escape situations in which, because there are no good positive examples to use next, it is difficult to make progress through positive reinforcement.

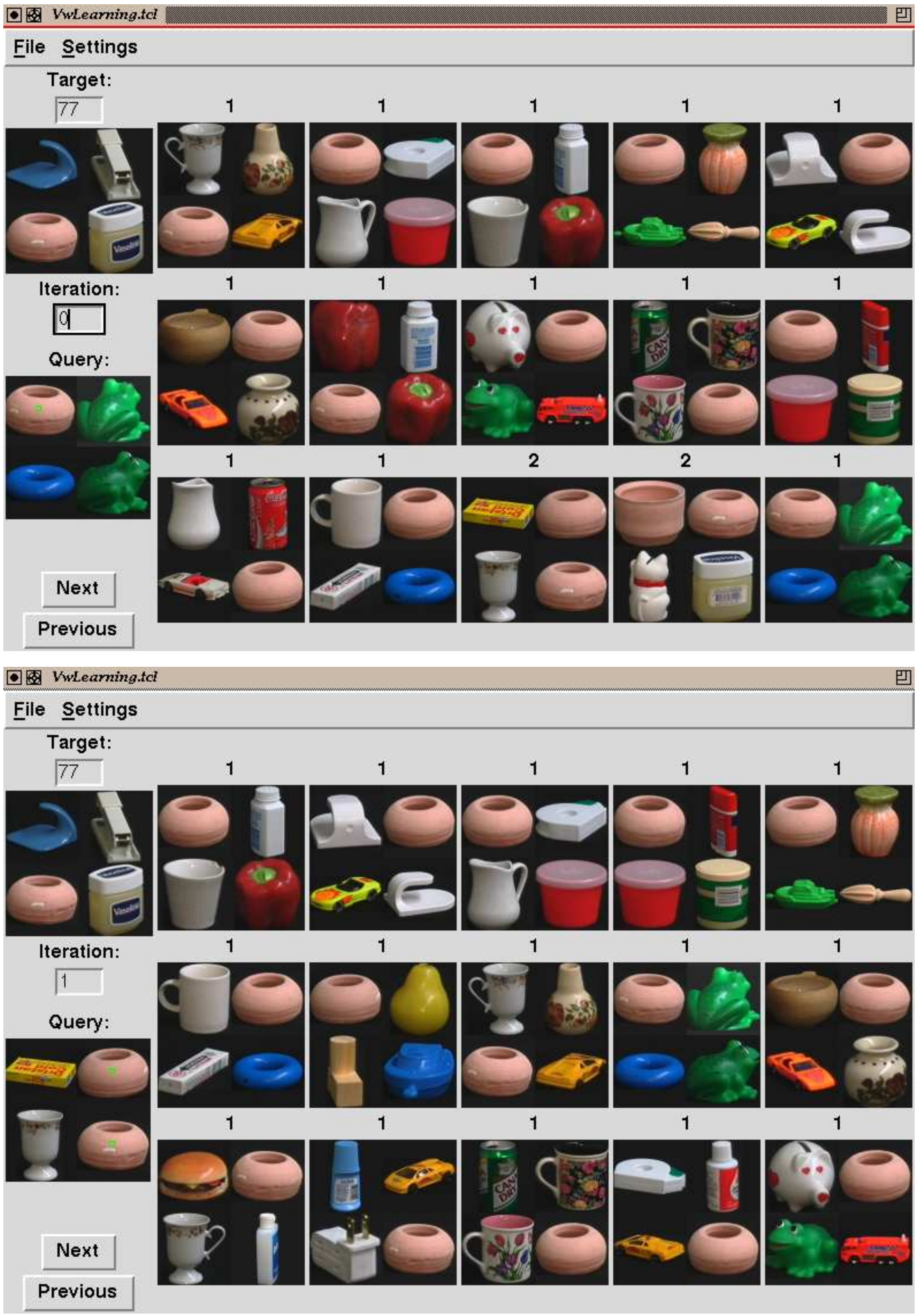

Figure 7.12: Two iterations of positive feedback.

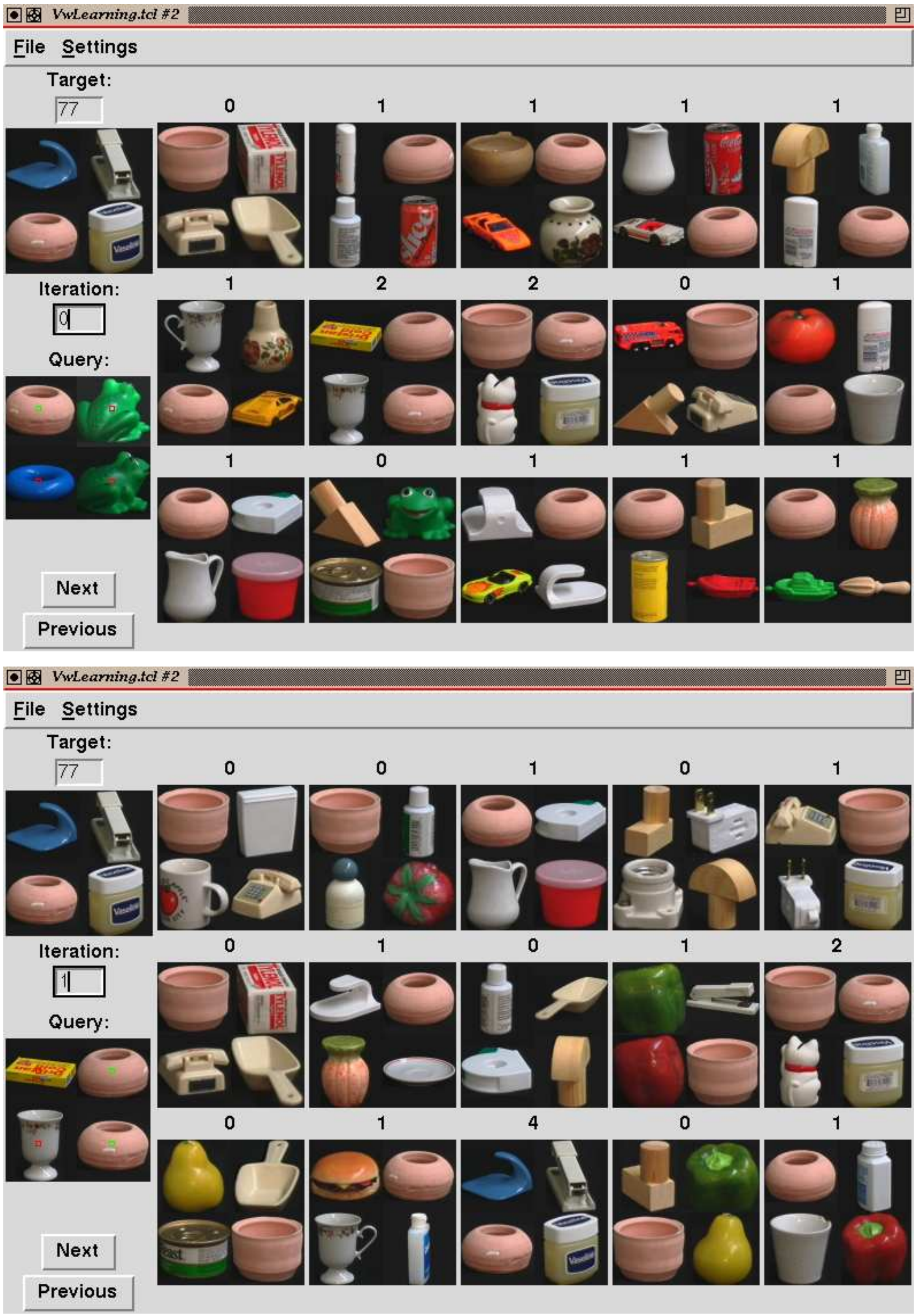

Figure 7.13: Two iterations of the query of Figure 7.12 but now allowing both positive and negative feedback.