Hunting moving targets: an extension to Bayesian methods in multimedia databases

Wolfgang Müller, David McG. Squire, Henning Müller and Thierry Pun

Computer Vision Group, University of Geneva

24 Rue du Général Dufour, CH-1211 Genève 4, Switzerland

ABSTRACT

It has been widely recognised that the difference between the level of abstraction of the formulation of a query (by example) and that of the desired result (usually an image with certain semantics) calls for the use of learning methods that try to bridge this gap. Cox *et al.* have proposed a Bayesian method to learn the user's preferences during each query.

Cox *et al.*'s system, `PicHunter`,[1] is designed for optimal performance when the user is searching for a fixed target image. The performance of the system was evaluated using target testing, which ranks systems according to the number of interaction steps required to find the target, leading to simple, easily reproducible experiments.

There are some aspects of image retrieval, however, which are not captured by this measure. In particular, the possibility of query drift (*i.e.* a moving target) is completely ignored. The algorithm proposed by Cox *et al.* does not cope well with a change of target at a late query stage, because it is assumed that user feedback is noisy, but consistent.

In the case of a moving target, however, the feedback is noisy *and* inconsistent with earlier feedback.

In this paper we propose an enhanced Bayesian scheme which selectively forgets inconsistent user feedback, thus enabling both the program and the user to "change their minds". The effectiveness of this scheme is demonstrated in moving target tests on a database of heterogeneous real-world images.

Keywords: relevance feedback, query drift, target testing, Bayesian methods, user modelling

1. INTRODUCTION

With large unannotated image and video collections becoming more and more common, there arises the need for tools which perform content based image retrieval. Much attention has been given to systems supporting the user in finding images similar to one example, enabling the user to give feedback.[2,3] Usually the problem of finding a useful seed has been ignored. Most of the systems give the user the possibility to get random choices of images in the database. This is suitable for testing query methods, but clearly insufficient, if one actually wants to *use* the database for real queries. What one needs here is a process, which provides a suitable seed.

However, little attention has been paid to systems which support the browsing process,[1,4] *i.e.* a process in which the user moves freely through feature space, by expressing his or her preferences. To our knowledge these systems were specialised browsers without nearest-neighbour capabilites. This paper focuses on the browsing process and puts it in context with query by example, showing that these capabilities are complementary.

The method for measuring performance of browsing systems established in[1] is the target test: the user is shown an image, which he or she then has to find in the database. The number of interaction steps is recorded. A good system minimises the number of interaction steps necessary for finding an image.

One point of criticism against the target testing method is its failure to capture the often gradual, sometimes sudden, changing of mind users experience during longer query sessions. In this paper we describe *TrackingViper*, a system which takes the possibility of query drift into account. Along with it we propose an extension of the target

{`Wolfgang.Mueller|David.Squire|Henning.Mueller|Thierry.Pun`}`@cui.unige.ch`
tel. ++41 22 705 7628, fax. ++41 22 705 7780

test which permits the measurement of the performance of systems which try adaptively to help the user in the browsing process.

In the following section 2 we describe the Bayesian method used by `PicHunter` as well as our modifications to its model, which rely on a model of user feedback inconsistency: if the user gives feedback which is strongly inconsistent with earlier judgements, this can be caused by a change in what he or she is looking for. Different models will be described.

It follows section 3 which establishes a new benchmark for the performance of image databases: the moving target test. It is put in context with other performance measures.

2. BAYESIAN METHODS FOR BROWSING QUERIES

2.1. `PicHunter`: a static target searching system

Information retrieval systems usually try to find (more or less explicitly) documents δ which have a high probability of being wanted by the user given the query. Here the query can be text as well as an example. In most systems the user is given the chance to improve the result by giving feedback, thus looking for δ which optimise $p(\delta$ wanted by the user|query, feedback). This paradigm has been largely adopted by the image database community. In image databases, the situation however is quite different, due to the inability of the user to formulate pictorial queries as precisely as a textual or a SQL query. In many cases the goal will be to find one or more interesting images in the collection, without being able to furnish a suitable seed for a query by example. This leads to the browsing query case, in which an explicit query does not exist, and in which we look for images which optimise $p(\delta$ wanted by the user|feedback over several steps). The main difference from the normal IR case is that relevance feedback is not seen as an improvement to an already well-phrased query, but as a query language for a user who is unable to formulate ad-hoc queries.

Cox *et al.* try to solve this rephrased query problem by maintaining a probability distribution which contains for each document δ of the database the probability that δ is the target of the query, *i.e.* the goal the user has in mind when starting the query. To this end the system, `PicHunter`, gives at each interactive step a small set of images, the suggestion S, to the user. The user responds by giving some feedback F to the system. The update is done using the classic Bayesian rule:

$$p(T=\delta|F,S) = \frac{p(F|T=\delta,S)p(T=\delta,S)}{p(F,S)} \tag{1}$$

In this formula $p(F|T=\delta,S)$ is the user model: it describes the expected feedback, if we know that the target is δ and the suggestion S had been given. $p(T=\delta,S)$ is the prior knowledge of the target's whereabouts, and $p(F,S)$ (the probability that F is given at the same time that S is suggested) is a normalising factor. The suggestions S are chosen to minimise the expected number of comparisons needed to find the target using an entropy argument. The feedback given by the user is given by comparison, *i.e.* the user decides which of the suggested images are closer to the target than other suggested images. Thus the elementary feedback a user can give is one comparison between two images: we write

$$\mathit{user}(T:i,j) \quad :\Leftrightarrow \quad \text{The user considers } i \text{ closer than } j \text{ to the target } T \tag{2}$$

User modelling in `PicHunter` is based on the assumption that one is in the possession of a distance metric $d(i,j)$ for images i and j, which captures the human "internal" similarity measure to such an extent that one can assume that it is perfect except for blurring by "mistakes" of the user:

$$p(F|T) = \frac{1}{1+\mathrm{e}^{\frac{d(T,j)-d(T,i)}{\sigma}}} \tag{3}$$

where σ is a free parameter which has to chosen before the process by the implementor. In the case of $\lim_{\sigma\to 0}$ (3) converges to

$$p(F|T) = \begin{cases} 1 & d(T,i) < d(T,j) \\ 0.5 & d(T,i) = d(T,j) \\ 0 & d(T,i) > d(T,j) \end{cases} \tag{4}$$

This limit case of no blurring shows two potential weaknesses of this method:

1. σ obviously depends on the image collection as well as the intended users. More importantly, it depends also on the metric employed for the search.
2. Once an image δ is discarded from the set of potential targets (*i.e.* $p(T = \delta) \ll 1$) by "wrong" feedback, it is difficult for δ to be reconsidered.

These two points are equally important for the moving target problem which we address in this paper.

2.2. *TrackingViper*: modelling the changing mind

2.2.1. Motivation

The shortcomings of the `PicHunter` can be summarised by saying that it would be desirable to take our uncertainty regarding the user model into account, using stronger methods than blurring. This need is emphasised by another observation: In the beginning of a real–world browsing query the user usually has a target in mind. During the query, however, the user often changes his or her target as a function of what he or she thinks can be found in the database.

One method for coping with this is to provide the user with an explicit means to express changes of mind. However, this would place the burden on the user to decide clearly, if he or she changes his or her ideas about the target or not, and how much these changes affect user feedback given in previous steps.

We propose here a framework which consists of weighting the different comparisons according to the degree to which they are trusted. Comparisons which are in less contradiction with others are more trusted.

Consider the simple case of a modified Hi-Lo game (the Hi-Lo game is classic programming exercise for beginners: let the user find a real number in a given interval). Consider finding a number t in the interval between $[0, 1]$. One person, A, asks B which one of the points x_1, x_2 is closer to t. If the B is good at arithmetics, there is no hesitation: A has to choose x_1 and x_2 such, that they are equidistant from the midpoint. By doing this, A can expect to half the set of points still to be considered. The problem will be reduced to either finding a point in $[0, 0.5)$ or $(0.5, 1]$. However, if you consider A and C, where C sometimes has little problems with addition subtraction and comparing numbers, A will either employ the same tactics as `PicHunter` *i.e.* "blurring" and/or A will deliberately choose x_1 and x_2 so that "surprising" results can be detected.

This observation leads us to a tradeoff: a sequence of comparisons chosen so as to strictly minimise the number of steps to be taken will each time halve the distribution, but it will not allow for the detection of inconsistencies except at the moment where there are no more points to consider. A "no-surprises" sequence of comparisons, however, will start at one end of the interval and consider every point.

2.2.2. A framework for the definition of inconsistent user behaviour

For our definition of inconsistence of user feedback, we regard it as convenient to express (1) rather as an intersection of probabilistic sets. A probabilistic set over a set of items I, $\mathcal{I}$, is a set of pairs (p, i) where $p \in [0, 1]$ denoting the probability of an item that an $i \in \mathcal{I}$ is in the set.

$$\mathcal{I} \quad := \quad \{(p, i) | i \in I \text{ and } p \in [0, 1]\} \tag{5}$$

We define now the intersection between two probabilistic sets $\mathcal{I}$, $\mathcal{J}$ with weighting constant w (to be explained below), using the function f as

$$\cap (\mathcal{I}, \mathcal{J}, w, f) := \{(f(p_{\mathcal{I}}, p_{\mathcal{J}}, w), i) | (p_{\mathcal{I}}, i) \in \mathcal{I} \text{ and } (p_{\mathcal{J}}, i) \in \mathcal{J}\} \tag{6}$$

We write the multiplication of the probability of each element in a probabilistic set $\mathcal{I}$

$$\times (\alpha, \mathcal{I}) := \{(\alpha p, i) | (p, i)) \in \mathcal{I}\} \tag{7}$$

We can write the normalisation (in the probability distribution sense of the word) of a probabilistic set, as

$$\text{Normalise}\,(\mathcal{I}) := \times\left(\frac{1}{\sum_{(p,i)\in\mathcal{I}} p}, \mathcal{I}\right) \tag{8}$$

If the suggestion was S, the user feedback F and the knowledge prior to a learning step is described by the probabilistic set I, the learning step (1) becomes

$$J \quad = \quad \cap\left(\mathcal{I}, \{(p(T = X|S,F), X)|X \in \mathit{database}\}, 1, \mathit{multiply}\right) \tag{9}$$

with $\mathit{multiply}(a,b,c) = a \cdot b \cdot c$.

We will call $\{(p(T = X|S,F), X)|X \in \mathit{database}\}$ the *feedback set* of F (and T):

$$\text{Feedback}\,(F,S,T) \quad := \quad \{(p(T = X|S,F), X)|X \in \mathit{database}\} \tag{10}$$

Thus we can regard the learning process as a sequence of intersections of probabilistic sets. The probabilities p of the members (p,i) of the resulting sets represent a plausibility of i being the target. Accordingly we define: a weighted set of user feedback steps F_k with weighting constants w_k, $\{(w_1,F_1),...,(w_N,F_N)\}$ is consistent under a plausibility predicate Pred (given the combination function f) iff

$$\exists(p,i) : \text{Pred}(p,i) \text{ and } \left((p,i) \in \bigcap_{k\in\{1,...,N\}} (\text{Feedback}\,(F_k,S_k,T)\,, w_k, f)\right) \tag{11}$$

i.e. there is at least one image in the intersection of the feedback sets which is considered to be plausible by the predicate Pred. For short, we write $\text{Consistent}\,(\{(w_1,F_1),...,(w_N,F_N)\})$ iff$\{(w_1,F_1),...,(w_N,F_N)\}$ is consistent.

The w_k in equation (11) correspond to our "belief" into $\text{Feedback}\,(F_k,S_k,T)$ as result of a useful comparison. Our learning problem is thus to be augmented by the search for w_k which make (w_k,F_k) consistent. If one wants to view this in a more Bayesian style, w_k modifies the user model, depending on the belief in the comparisons. This leaves us the choice of a function which finds for the F_k proper w_k. Furthermore, we have to choose a combination function f and the user model.

First of all we want to emphasise the temporal component: we assume that more recent comparisons are more credible than older ones. Without this assumption, our relations become symmetric, and we have no way of deciding which comparison is to be weighted lower if two comparisons are inconsistent.

Knowing that computing time is limited, one possibility is to make each w_k a function of the feedback given in the m previous and m following steps $(F_{k-m}, F_{k-m+1}, ..., F_{k+m-1}, F_{k+m})$ for some small m. In this paper we take a more rule based approach given by the following algorithm:

N is the number of feedback steps given so far.

1. $i \to N$
2. $w_i \to \begin{cases} 1 & \text{Consistent}\,(\{(w_1,F_1),...,(w_{i-1},F_{i-1})\}) \\ 0 & \text{otherwhise} \end{cases}$
3. until $(i = 1$ or $\#\{w_k|k \in \{i,...,N\}$ and $w_k = 0\}) > \vartheta$ do $i \to i-1$ and goto 2

This leaves us with the choice of a suitable Pred, the choice of ϑ, as well as the choice of a combination function f. The Pred is a simple threshold comparison. In this paper we chose $f(a,b,c) = \mathit{multiply}(a,b,c)$.

The user model was chosen as

$$p(F|T) \quad = \quad \begin{cases} 1 & d\,(T,i) \le d\,(T,j) \\ 0.1 & d\,(T,i) > d\,(T,j) \end{cases} \tag{12}$$

Similar to the original papers about `PicHunter`, we estimated the expected entropy gain by sampling several suggestions S_i from the current distribution, and taking the S_i which maximised the decrease of entropy in the current distribution. In order to generate suggestions which enable inconsistent user feedback, we slightly favour S which minimise $\sum_{k=1}^{N} w_k$, *i.e.* which force forgetting of user feedback.

2.2.3. The system

The tests in the present work were performed using *Viper*,[3] a system which uses techniques inspired by text retrieval (inverted files), on a very large quantised feature space. *Viper* obtains in Query-By-Example good Precision-Recall in interactive time for databases of moderate size ($\mathcal{O}(10000)$ images). The features have proven to be highly selective, and in relatively good accordance with human perception.

Viper's features and scoring algorithms yield a distance measure which is not symmetric. This is in accordance to psychophysical evidence, that human similarity perception is not symmetric.[5] However, for use in the present framework, asymmetry of the distance measure is undesirable. The distance measure is also required to fulfil the triangle inequality. A symmetric distance matrix fulfilling these constraints to a suitable extent was built for the experiments.

Viper is designed for flexibility. Our present system allows the use of browsing queries and nearest neighbour queries in parallel. This capability was used for some of our tests.

3. EVALUATION

The normal QBE process corresponds to deep exploration of the feature space in the immediate neighbourhood of the example. The relevance feedback usefully leads to a deformation of the feature space in order to better capture the user's view of similarity, and to capture the differences between the user's example and the user's information need. How much the feature space chosen for a program corresponds to the average information need of the unexperienced user can be measured in precision–recall plots and derived measures, as well as using other methods using user relevance data.

Target testing, however, captures the *mobility* of the user within the feature space. He is supposed to move in an autonomous fashion in feature space, thus finding an image whose characteristics are not known to the database. In this target testing and query by example are testing two complementary properties which should be present in every CBIRS: Browsing query systems help in *query formulation*, while good QBE systems have a good query performance.

In most real cases the user will be rather interested in approaching the target using a browsing query mechanism. After a certain point however, he or she will profit from a good and quick overview over the images which are inside a certain region of the feature space which can be obtained using a nearest neighbour query.

The weakness of the target testing method, however, is its focus on one target image. This generates mainly two problems:

- arguably, the user will react in a different way, if knowing the whole image to be found, than if he or she knows only a semantic category the wanted image falls in
- equally arguably, a real user will change his or her mind during the browsing process. Giving a target image to the test person is like telling the buyer which jeans to buy before sending him or her into the shopping mall and guaranteeing that he or she will find it in one of the businesses in the mall.

We consider that giving a target image to the user is the one method to verify that different results for different systems are not only due to the fact that the test persons are more or less demanding. However, we suggest to model the inevitable change of mind by giving the user a sequence of targets he or she has to visit. This simulates the moves of the target and the ability of the system to follow moves and to detect abrupt changes of the target. This method of testing we call *moving target test*.

In the experiments described in the following subsections we give some moving target tests, for "simple" images taken from a complex real–world database containing 2500 images (TSR2500) provided by *Télévision Suisse Romande*, the broadcasting corporation of the French speaking part of Switzerland. Before the presentation of these results we give a short summary of simulations which capture the "best case".

3.1. Simulation

We simulate a user who uses exactly the same distance metric as the program. This user tries to find a sequence of four images in the database. The number of tries for each retrieval is counted.

Of course, these simulations do not prove anything about real users. However, they show, that the requirements for the use of this method are met by the distance measurement, combination function and algorithm. If the user does not make any "mistakes", he or she will be able to find a sequence of images. Because of the perfection of the simulated user the simulations should provide an upper limit for the performance of this method.

We did two kinds of simulations:

1. a simulation, in which the simulated user gave at each step negative feedback for all images but the best match to the target, giving positive feedback for the best match. Here the simulated user had to see 38 images on average for reaching each query target.

2. a simulation in which it gave negative feedback to the worst match and positive feedback to the best match. Here the simulated user had to see 75 images on average for reaching each query target.

Simulations of a *TrackingViper* without forgetting (this is close to `PicHunter` with $\sigma = 0$) suggest that without forgetting feedback, performance seriously degrades after finding the first target.

3.2. User Experiments

For moving target tests we chose four scenarios:

***Viper* with and without feedback memory** for previous feedback steps: As a reference, *Viper* was used with a random seed to find the targets of the target sequence. 20 images were visible at each feedback step.

***TrackingViper*:** *TrackingViper* memorised the last 10 feedback steps and cumulated the knowledge gained from older feedback in an additional eleventh feedback set. ϑ was ignored. At each step 5 images were shown.

Split screen: In addition to the suggestions (5 images) provided by *TrackingViper* 15 images from a *Viper* nearest neighbour query were shown. Using this, the user had the opportunity to explore the feature space given by the feedback images chosen by the user. The fact that 15 images were shown from a nearest neighbour query, and only 5 from a suggestion, is not a contradiction. The choice of a suggestion takes time linear to its size, while a nearest neighbour query scales much more favourably.

As a general modification to the original `PicHunter` papers, in our experiments, the user had explicitely to give both positive and negative feedback, thus leaving space for indifference in case of uncertainty of the user.

Choice of test images: Preliminary experiments had shown that the TSR2500 database contains many very small clusters of images, which are so semantically different from each other that it is difficult for the user to judge image similarity (Geman *et al.*[6] write in this context of "virtually random" comparison outcomes). For the test in this paper we chose four images from large clusters in the database: banknotes, trademarks and flags, sunsets (very dark background), airplanes (mostly sky).

Test persons two test persons with image processing background performed multiple experiments with *TrackingViper* and modified versions. As it will be described and discussed below we had learning effects during some of the tests.

***Viper* without feedback memory** The user was started from a random set of 20 images, giving feedback in order to move in the direction of the next target. Here the user was able to give feedback at each step. This feedback was taken into account for the calculation of a new set of 20 images resembling the given feedback and then forgotten. 420 ± 100 images had to be seen by the user for finding all images in each target sequence. The details can be found in table 1.

Image	Test 1	Test 2	Test 3	Test 4	Test 5
$1 bill	80	60	120	100	100
Honda	40	80	80	140	140
Sun	60	120	120	260	120
747	140	60	80	100	100
Sum:	320	320	400	600	460

Table 1. This table summarises the outcomes of 5 moving target tests (Test 1 through 5) with *Viper* without feedback memory. The test consisted in finding a one dollar bill, a Honda logo, an image of the sun, and an image of a flying Boeing 747 in this sequence. Each time the user started at a different random state. In each cell of the table the number of images seen to find the next target is noted (*e.g.* in the first test, the user needed to see 140 images, to find the 747 after having found the sun image.) The complete number of images seen in each complete moving target test is summed up in the last column.

***Viper* with feedback memory** Here the user also started with a random set of 20 images, giving feedback in order to move in direction of the next target. In contrast to the method described in the paragraph above, the user was able to cumulate feedback over several steps.

We did two runs of five tests with the same expert user. We observed, that the user learned quickly, how to optimise his target testing performance when using *Viper*. He was able to reduce the number of images seen before finding the last target by approximately a third.

On first use by the user, 420 ± 140 images had to be seen by the user for finding all images in each target sequence. The details can be found in table 2.

Image	Test 1	Test 2	Test 3	Test 4	Test 5
$1 bill	120	100	140	100	100
Honda	80	80	60	220	80
Sun	80	40	440	80	40
747	20	120	40	80	100
Sum:	300	340	680	480	320

Table 2. This table summarises the outcomes of 5 moving target tests (Test 1 through 5) with *Viper* with feedback memory. The test task was exactly the same as in table 1.

After having gained some experience, only 260 ± 40 images had to be seen by the user for finding all images in each target sequence. The details can be found in table 3.

Image	Test 1	Test 2	Test 3	Test 4	Test 5
$1 bill	100	120	100	100	60
Honda	100	40	60	80	80
Sun	80	80	20	60	40
747	40	40	40	40	40
Sum:	320	280	220	280	220

Table 3. This table summarises the outcomes of 5 moving target tests (Test 1 through 5) with *Viper* with feedback memory. The test task was the same as in table 1.

TrackingViper Five experiments with an identical target sequence were performed, starting at different starting points. The user needed 65 ± 11 iterations for performing the task; the user therefore scanned on average 82 images in 16.5 iterations before finding a target of the target sequence. This is approximately 15 times better than chance. While these results are quite satisfactory (better than to *Viper* before learning) and surprisingly close to the simulation results, the user found his situation quite difficult: images chosen by the system are not necessarily close to the images marked positive by the user. Often this is desired. Sometimes, however, the user has the impression that his or her feedback had been "misunderstood".

320 ± 55 images had to be seen by the user for finding all images in each target sequence. The details can be found in table 4. Within the experiment the performance, attained by the test user using *TrackingViper* did not increase.

Image	Test 1	Test 2	Test 3	Test 4	Test 5
$1 bill	110	85	75	130	85
Honda	105	75	40	135	105
Sun	100	25	85	50	105
747	45	45	140	80	50
Sum:	360	230	340	395	310

Table 4. This table summarises the outcomes of 5 moving target tests (Test 1 through 5) with *TrackingViper*. The test task was exactly the same as in table 1.

Surprisingly, for the chosen queries, the simulation results are only slightly better than those of a skilled human user. This might be explained by the fact that a human user can consciously induce small contradictions, if he or she does not like the current suggestion. The system then will open up the distribution by forgetting parts of the old feedback.

Split screen *TrackingViper/ Viper* This method was a reaction to the subjective impressions of our test user when performing *TrackingViper* queries. At each iteration 20 images were shown to the user. 5 of them were a suggestion by *TrackingViper*, 15 the result of a nearest neighbour query, which used the feedback given in the last step. As one can see, these results are clearly the best of our tests.

Also the subjective impression when using this version was satisfactory: the *TrackingViper* part provide the user with good seeds which in the present simple setting were quickly usable for successful nearest neighbour queries.

On average, only 225 ± 30 images had to be seen by the user for finding all images in each target sequence. The details can be found in table 5.

Image	Test 1	Test 2	Test 3	Test 4	Test 5
$1 bill	80	20	60	140	100
Honda	40	20	60	60	40
Sun	60	40	20	20	20
747	60	100	60	40	80
Sum:	240	180	200	260	240

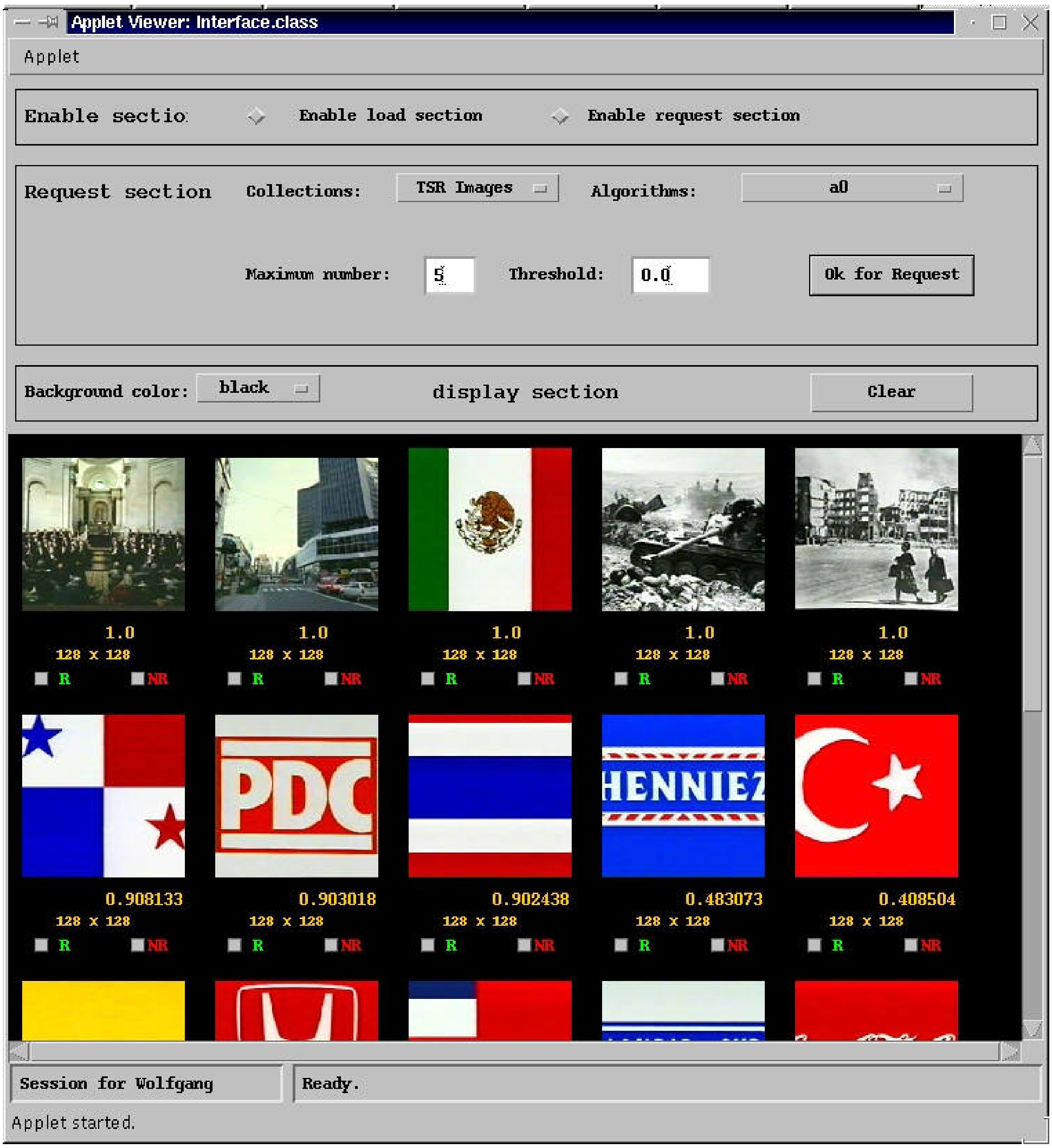
Table 5. This table summarises the outcomes of 5 moving target tests (Test 1 through 5) with *TrackingViper*. The test task was exactly the same as in table 1. These results are the best over all tested methods.

4. CONCLUSION

In this paper we presented an approach to tracking query drifts in target searches. In target searches the user tries to find an item (target) in the database starting from a small random sample.

The approach was implemented in the system *TrackingViper*. In order to measure the performance of the system we made user experiments with expert users. They used the system to find a number of targets in a fixed sequence. Targets were simple images taken from a dataset of 2500 images. Future tests will involve random targets, as well as non-expert users.

Our user experiments gave two main results:



Applet Viewer: Interface.class

Applet

Enable sectio ◇ Enable load section ◇ Enable request section

Request section Collections: TSR Images Algorithms: a0

Maximum number: 5 Threshold: 0.0 Ok for Request

Background color: black display section Clear

1.0 128 x 128 R NR | 1.0 128 x 128 R NR | 1.0 128 x 128 R NR | 1.0 128 x 128 R NR | 1.0 128 x 128 R NR

0.908133 128 x 128 R NR | 0.903018 128 x 128 R NR | 0.902438 128 x 128 R NR | 0.483073 128 x 128 R NR | 0.408504 128 x 128 R NR

Session for Wolfgang

Ready.

Applet started.

Figure 1. The last step, when trying to find the second of our four test images for the split screen (*Viper*/*TrackingViper*). The first line shows images suggested by *TrackingViper*, the next lines show the results of a nearest neighbour query on the feedback given in the last step. In the third row on second position, you can see a part of the target image.

1. *TrackingViper* enables the user to move in feature space by giving feedback. *TrackingViper* is able to follow changes of the user's wishes by forgetting parts of the user feedback deemed to be inconsistent with newer user feedback. Moving to a new target from and old, found target seems not to be more costly than explicitly starting a new query.

2. A combined system of a target searching system and a nearest neighbour QBE system was preferred by the user and performed best in our test. It enables the user to choose quickly between browsing movement in the feature space and intensifying his or her search in some point of the feature space.

Our results suggest that one should not see target searching systems as a stand-alone method, but rather as a convenient, efficient way of solving the problem of finding a suitable *seed* for nearest neighbour (QBE) queries.

In the future we want to explore more of the parameters given in section 2.2. At present we have the subjective impression that our system forgets too late, and sometimes too completely. We are especially interested in a method which does not have any parameters which have to be chosen before the query process. Furthermore, we would like to adapt our target searching methods so that they scale better with the database size than does the present explicitly maintained probability distribution. The results given in section 3.2 for "*Viper* with feedback memory" encourage this approach.

REFERENCES

1. I. J. Cox, M. L. Miller, S. M. Omohundro, and P. N. Yianilos, "Target testing and the `PicHunter` Bayesian multimedia retrieval system," in *Advances in Digital Libraries (ADL'96)*, pp. 66–75, (Library of Congress, Washington, D. C.), May 13–15 1996.
2. T. Minka, "An image database browser that learns from user interaction," Master's thesis, MIT Media Laboratory, 20 Ames St., Cambridge, MA 02139, 1996.
3. D. M. Squire, W. Müller, H. Müller, and J. Raki, "Content-based query of image databases, inspirations from text retrieval: inverted files, frequency-based weights and relevance feedback," in *The 10th Scandinavian Conference on Image Analysis (SCIA'99)*, (Kangerlussuaq, Greenland), June 7–11 1999.
4. J. Vendrig, M. Worring, and A. W. M. Smeulders, "Filter image browsing: Exploiting interaction in image retrieval," in *Third International Conference On Visual Information Systems (VISUAL'99)*, D. P. Huijsmans and A. W. M. Smeulders, eds., no. 1614 in Lecture Notes in Computer Science, pp. 147–154, Springer-Verlag, (Amsterdam, The Netherlands), June 2–4 1999.
5. A. Tversky, "Features of similarity," *Psychological Review* **84**, pp. 327–352, July 1977.
6. D. Geman and R. Moquet, "A stochastic feedback model for image retrieval," technical report, Ecole Polytechnique, 91128 Palaiseau Cedex, France, 1999.
7. T. V. Papathomas, T. E. Conway, I. J. Cox, J. Ghosn, M. L. Miller, T. P. Minka, , and P. N. Yianilos, "Psychophysical studies of the performance of an image database retrieval system," in *Human Vision and Electronic Imaging III*, B. E. Rogowitz and T. N. Pappas, eds., vol. 3299 of *SPIE Proceedings*, pp. 591–602, July 1998.
8. I. J. Cox, M. L. Miller, T. P. Minka, and P. N. Yianilos, "An optimized interaction strategy for bayesian relevance feedback," in *Proceedings of the 1998 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pp. 553–558, (Santa Barbara, California, USA), June 1998.
9. D. M. Squire, W. Müller, and H. Müller, "Relevance feedback and term weighting schemes for content-based image retrieval," in *Third International Conference On Visual Information Systems (VISUAL'99)*, D. P. Huijsmans and A. W. M. Smeulders, eds., no. 1614 in Lecture Notes in Computer Science, pp. 549–556, Springer-Verlag, (Amsterdam, The Netherlands), June 2–4 1999.