Evaluating Image Browsers Using Structured Annotation

Wolfgang Müller, Stéphane Marchand-Maillet, and Henning Müller
Computer Vision Group, CUI, University of Geneva, 24, rue du Général Dufour, CH-1205 Geneva, Switzerland

David McG. Squire
School of Computer Science and Software Engineering, Monash University, 900, Dandenong Road, Caulfield, Victoria 3415, Australia

Thierry Pun
Computer Vision Group, CUI, University of Geneva, 24, rue du Général Dufour, CH-1205 Geneva, Switzerland

In this article we address the problem of benchmarking image browsers. Image browsers are systems that help the user in finding an image from scratch, as opposed to query by example (QBE), where an example image is needed. The existence of different search paradigms for image browsers makes it difficult to compare image browsers. Currently, the only admissible way of evaluation is by conducting large-scale user studies. This makes it difficult to use such an evaluation as a tool for improving browsing systems. As a solution, we propose an automatic image browser benchmark that uses structured text annotation of the image collection for the simulation of the user's needs. We apply such a benchmark on an example system.

Introduction

Content-Based Image Retrieval Systems (CBR) are designed to help their user in finding images and making use of the visual content of every image in the collection, as opposed to labels attached to the images. The existence of large, yet unannotated image collections as well as the inherent limitations of image annotation motivate the research in this area.

Most current CBRs provide Query By Example (QBE). Here, the user gives one or more positive and negative example images to describe the images he or she would like to retrieve using the CBR. The system then presents a list of images to the user who usually has the possibility to refine his or her query, by giving additional examples from the response set. Techniques for the evaluation of such systems are close to those used in text retrieval. An overview of such techniques adapted to the case of image retrieval is given in Müller et al., (2000a).

While QBE addresses the question of how to find images similar to a given small set of images, interactive browsing addresses the need for finding a given image within a collection. In other words, QBE addresses the problem of closely exploring a given point (or a small region) in the collection, whereas browsing systems address the *mobility* within the collection.

There are two main technical directions of research in image browsing, leading to *deterministic* and *stochastic* browsing systems. Both of them present to the user successively refined overviews of the collection. The user can then express his or her preferences by marking one or more—depending on the system—images as relevant or irrelevant with respect to the goal of the search. This information is then processed, leading to a new, refined overview of the collection.

The differences between deterministic and stochastic systems lie in the way the overviews are provided, and in the way one can navigate through the image collection. *Deterministic* systems provide a hierarchy which guides the image search performed by the user. The hierarchy usually is precalculated. This drawback is at the same time an advantage: each image search will start with the same initial selection. The browsing process could be compared to moving through a city without a map. The user has the possibility to move through the collection using fixed paths and can memorize which images will lead to which other images during the search.

In contrast to this, *stochastic* systems provide overviews chosen by the system in response to the user-feedback. Typically multiple steps of user-feedback are taken into account. In contrast to hierarchical systems one has the possibility to mark multiple images as being more or less relevant to the query. As a consequence, at each stage of the retrieval process the user has so many possibilities for feedback that a precalculation of the possibilities is infeasible. Thus the task varies with each image search, and it is too complex to attempt a brute-force calculation, which leads to the use of Monte-Carlo[1] methods implying reproducibility *on average,* as opposed to *exact* reproducibility.

Both kinds of browsing systems have in common that a true test of their performance requires interaction with a user: the test user is presented with a target image, which he or she tries to find using the system. The performance is measured in terms of numbers of images the user had to look at. Images encountered twice are counted twice.

To our knowledge only one deep test of this kind has been done, the test of PicHunter as described by Cox et al., (1996) and Papathomas et al., (1998). In fact, for research groups of small size and low financial resources, tests like the one of PicHunter are difficult to conduct. Test users are hard to get, and it is difficult to evaluate the influence of the test user's background for the experiment (computer vision researchers make systems look better than novices, but by how much?). Moreover, in deterministic systems, users cannot be asked twice to perform the same test because they are likely to remember useful details of the last test run.

The first attempt at solving such testing problems was automatic benchmarking using low-level features (*i.e.* color and texture). The user is replaced by a piece of software which tries to find the target image. Cox et al. used this in PicHunter as a proof of concept backed up by real-user experiments. Vendrig et al., (1999) used this as the only benchmarking method for their deterministic browsing system. However, in both cases the benchmark uses the same user-model as the system to be tested, thus using the testing hypothesis («we have a useful feature set coupled with a useful learning method») for its own verification. Examples that illustrate the shortcomings of this approach are given below in the section about requirements for a good browser benchmark. It is argued that low-level feature based systems are not apt to function as user simulators for benchmarking other low-level feature based systems.

In this article, we advocate a browser benchmark which is based on structured annotation. The annotation is used to simulate the learning problem a browser is facing: closing the semantic gap between visual low-level features and the semantic concepts the user is looking for. The problem of finding a good annotation method for our purpose is nontrivial. In this article we describe the development of a structured annotation method coupled with an appropriate retrieval method for graphs with weighted edges and nodes.

Defining the Goal of Image Browsers

For defining a useful performance measure, one needs to first define what is optimal performance. In this article, we define a performance measure based on the simulation of user behavior; we also define which aspects of user behavior we want to simulate. Or, *in which aspects is the browsing system supposed to help the user?*

CBR Using Low Level Features

Content based image retrieval was invented as an enhancement to image annotation, and as an answer to the lack of annotation in common image collections. As the classical computer vision problem («tell me, what's in this image») remains unsolved, CBR make do with low level features, sometimes accompanied by optional annotation and sophisticated interaction techniques. Using annotation in images has been shown to improve retrieval performance of low-level feature based systems (Papathomas et al., 1998). This is unsurprising, because annotation contains the semantics we are not fully able to capture in low level features. However, the main issue for measuring the success of CBR research is *evaluating the contribution of low-level feature based systems to retrieval success.* As a consequence, we constrain the formulation of a benchmark for browsing systems on systems which do not use annotation for the search. This restriction provides the opportunity to use the annotation for the simulation of the user's wishes.

Formal Description of the Browsing Problem

In the following we assume that the user browses a given collection of images (of size N) to find one *target* image T. Derived problems (find one out of n images in a collection of size N) are usually easier, but not in any fundamental way. The user applies some kind of distance measure $d_{\text{semantic}}(I_1, I_2)$ between images I_1, I_2 which is mainly semantic-based. The browser, however, will apply a different distance measure, based on low level features $d_b(I_1, I_2)$. The discrepancy between these two measures is the consequence of the *semantic gap.* There are now two alternatives: either the browser's measure is close enough to the user's measure to permit browsing without having to traverse large parts of the collection, or the system tries to learn from the user's feedback a measurement $d^F_{\text{browser}}(I_1, I_2)$ which approaches d_{semantic} in a sufficient manner.

Testing the ability of an image browser to help the user in finding a given image in a given collection is called *target testing.* It was first described in Cox et al., (1996) and Smith and Chang, (1996).

Requirements for a Good Browser Benchmark

As stated, the main goal of an image browser is helping the user to close the semantic gap between low-level visual features and high-level semantics in order to browse through a collection in a way he or she understands. This we identify as the principal requirement which should be evaluated by an image browser benchmark.

[1] PicHunter and *TrackingViper* both pick the images shown to the user such that the expected number of query steps are minimized. To do the exact calculation would cause the algorithm to have very high complexity. Instead the expected number of query steps is calculated for a *small number of random selections,* and the best random selection is taken.

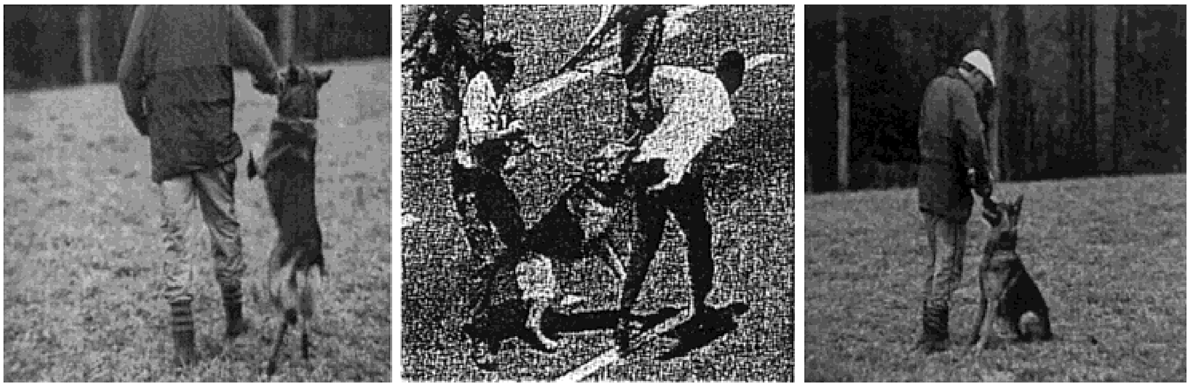


FIG. 1. An example where the semantically more distant image is considered closer to the query: Viper (Squire, Müller, & Müller, 1999) (in high-speed–low-quality mode) considered the middle image closer to the left image than the right image. This is due to matching the black trousers of the man in the middle picture to the dog jumping in the left picture. However, clearly the semantic of the right image is closer than the left.

As a consequence, *any* automatic benchmark which uses low-level features only is useless for true evaluation. The process of learning a mapping between two different color spaces is much easier than learning uniquely from user feedback that the user wants images with at least one dog in it (Fig. 1). Furthermore, such an evaluation masks the principal problem of image browsers: in many situations, meaningful answers are not possible without knowledge about the low-level feature set, as illustrated in figure 2.

If a browser uses both visual low-level features and textual features, it is impossible to evaluate the contribution of visual features to the query result. In this work we will focus on the ability of the browser to use visual features to bridge the gap between visual features and the semantic image content. Consequently, we limit our benchmark to systems, that *do not use textual annotation.*

Text annotation and low-level features also are separated by a semantic gap. This semantic gap might not be as large as the semantic gap between the user's wishes and low-level features. However, it is considerable and *similar in nature* to the semantic gap a user experiences. As a consequence we suggest benchmarking image browsers by evaluating their target testing (Cox et al., 1996) performance when simulating users using a textual distance measure d_{text}. The distance $d_{\text{text}}(I_1, I_2)$ between two images is determined using text retrieval techniques on the annotation of I_1 and I_2. Details are described below.

Ranked QBE on Structured Annotation

We now focus on a semantic-based distance measure between images. At first glance, this problem seems to be easily tractable using classic text-retrieval techniques. However, this is not the case, as described below.

We then follow with the structured annotation approach we adopted, along with the retrieval method we used on the annotation. We give a performance-evaluation of this annotation and compare it to the performance of a similar, unstructured approach.

Differences with Classical Text Retrieval

Textual information retrieval is an old area of strong economic interest. Much research has been done in the last 40 years. The successful establishment of a common benchmark by the Text REtrieval Conference (TREC, 1999) has

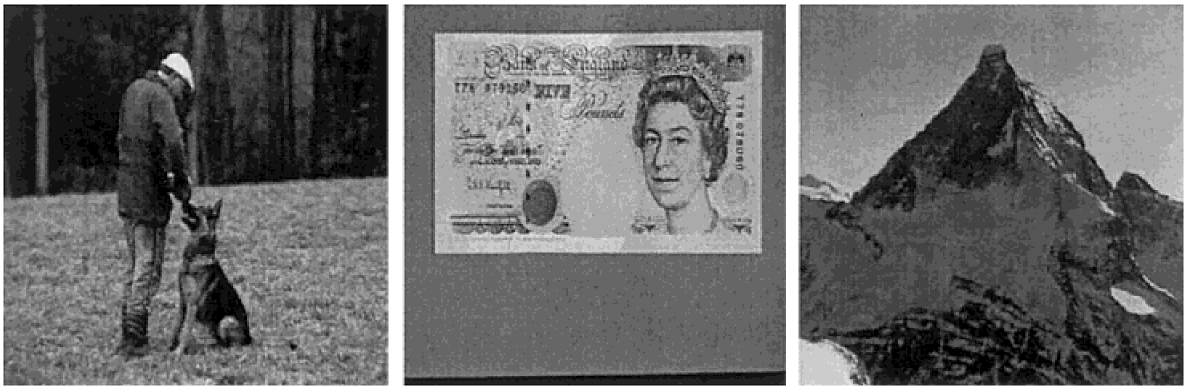


FIG. 2. An example that illustrates that in many cases, a sensible answer does not exist. During the browsing process, the user is often confronted with questions like the following: "What is more similar to the image of the man and the sitting dog: the mountain or the pound note?" Stochastic browsers provide the user with a possibility to decline an answer if the selection does not offer the possibility for sensible feedback.

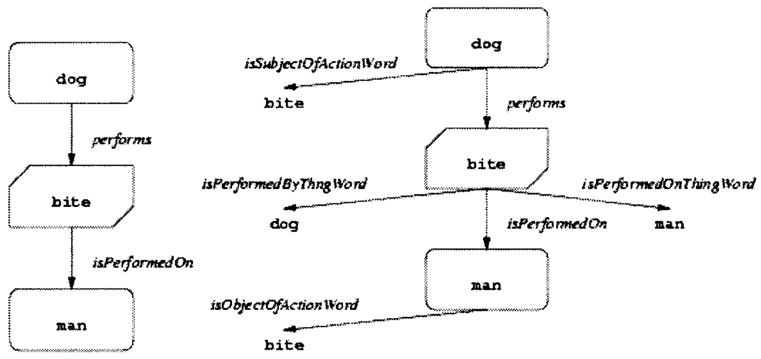


FIG. 3. A basic example of structured annotation as described in the text. Initially, the annotation is structured as shown in the image on the left. This structure is automatically augmented, as shown in the figure on the right. This becomes useful in inexact matching.

rendered results comparable and has created a general competition for the best text retrieval solution.

Presently, the systems performing best in TREC use very little linguistic or semantic knowledge and Natural Language Processing (NLP) techniques have had little success up to now. Vorhees (1999) identifies as one reason that the disambiguation techniques are too error-prone: the precision gained by more accurate modeling of the word relationships is lost by trusting too much in wrongly established word relationships.

However, the case of image annotation is different:

- TREC deals with documents of kilobytes in size whereas annotation usually is much shorter. As a consequence, statistical measures like the classic *tf.idf* measure will produce less accurate results.
- While chances are high that in well written long texts multiple synonyms of one meaning are used, usually only one synonym of a given word will appear in a very short text. Thus the analysis of short texts requires better determination of the true word sense.
- In the scenario where a query has been formulated by hand, one can assume each query term to be relevant to the user. However, in our case we are interested in the distance between documents, *i.e.* we are interested in QBE instead of hand-formulated queries. As a consequence, not every query term is relevant to the user who gave the example.
- Annotation is made for retrieval purposes, as opposed to natural language texts whose goal is primarily to convey information to a human reader. Adding structured annotation to an image takes only little more time than adding unstructured annotation to an image. As a consequence, we can use the structured annotation for replacing the faulty disambiguation step by hand-made disambiguation.

Consider the following example: A database contains an image, annotated by the caption `A dog jumping over a bar. Bushes in the background. People in the background.` Using this as a positive example in a QBE query on captions, a normal text query might retrieve `A statue. Bushes in the background. People in the background.` The result `A dog jumping over a bar.` would get a lower rank, as less items match with the query.

Intuitively, each item we want to describe has to be described using at least one word, even if it is of little importance. Because of the shortness of the annotation text, it is a matter of chance (*i.e.* the statistics of the database) if the term we employed for the background item is rare or frequent in the database; thus, which weight it will receive is very uncertain. In other words, there is a need for preweighting of terms. This can be derived from structured annotation, as described next.

The Structure of the Annotation

Structuring the annotation causes little overhead, as it was expressed in the last subsection. Structuring the annotation adds information about the importance of the different items of the annotation; such structuring is meta-annotation. Its main advantage for us lies in enabling the use of *a-priori* information about the importance of items in the annotation text.

In our annotation effort, we emphasized the importance of participation in an action. The rationale for this is that most of the time, if there is action in an image, the parts of the image that are not implicated in the action are less important and be considered as part of the background. Furthermore, we wanted to be able to express subject-object relations: classic text retrieval methods will not distinguish between `Man bites dog.` and `Dog bites man.` However, while the first example surely *is* a news item, the second happens every day.

We designed a small set of relationships, not with the intention of being linguistically complete, but with a view to our image collection and the interesting relationships between items in this collection. For structuring our annotations, we implemented a small language which compiles facts with Prolog-like syntax into Prolog. This permits the expression of simple semantic networks. The annotation presented here is not a full fledged linguistic annotation. However, it captures the basic relationships between items of the annotation. Please note that the annotator has to name by himself the nodes of the semantic net for one image, but that the burden of finding non-conflicting node IDs for the whole collection is taken care of by the compiler that translates this structure into Prolog. The language used here has the advantage that while being close enough to programming languages for testing purposes, the syntax is short and simple to memorize. The left half of Fig. 3 shows the semantic network derived from the annotation

```
actor (dog).
actor (man).
action (bite).
performs (§dog, §bite).       % read:
% “dog performs bite”
isPerformedOn (§bite, §man).  % read:
% “bite is performed on man”
```

The Retrieval Method

Efficient queries on conceptual graphs have been described in Ounis, (1999), Ounis and Pasca (1998). In the following we describe a similar efficient approach which is more easily extended towards the use of multiple classes of features while preserving the graph structure of the problem.

It is well known that graph matching is a difficult problem. However, in our situation, we can make use of the structural knowledge of the graph, and our knowledge about what interests us in the graph to simplify the problem. For matching two images, we take a three-step approach:

1. Identify similar nodes between query and match. Assign a score to each matching node.
2. Verify the relationships between pairs of similar nodes in query and match. Increase the score for each pair of similar nodes in similar relation.
3. Sum the score for each matched node. The result is the score for an image.

Clearly, the crucial step is the first one: without proper identification of similar nodes, the verification of the relation will be impossible. However, in our case, this identification is simple. We store with each actor the verb of the action it performs, the verb of which it is object. This means that we perform a simple, unstructured query for the features of each node that are connected to one node only (`modifies`, `enumerates`, `isSubjectOfActionWord`, *etc.*) which we use as a basis for a greedy identification of nodes with similar functions in the query and matching images. Then we use these results for verifying the relationships between nodes. As a consequence, this retrieval method approaches classic inverted-file text retrieval algorithms in efficiency.

Clearly, the features used for the unstructured query step can be any kind of features. This provides a way of tight integration of visual features into this framework.

Performance of the Annotation

Here we describe the performance of the annotation when doing QBE on an annotated collection. The goal is to show that this annotation gives a good «one-shot» retrieval performance, making the annotation suitable as the basis for the simulation of a real user in a browsing scenario. Our results are compared to the results of applying classical text retrieval methods on unstructured annotation with the same content. This unstructured annotation was derived from the structured annotation by removing the structuring elements from the structured annotation.

For this, the described annotation and retrieval scheme was used on the 500 images provided by the *Télevision Suisse Romande,* the French-speaking Swiss television station. This image collection was chosen for its diversity. It contains scenes of varying complexity and varying degree of action.

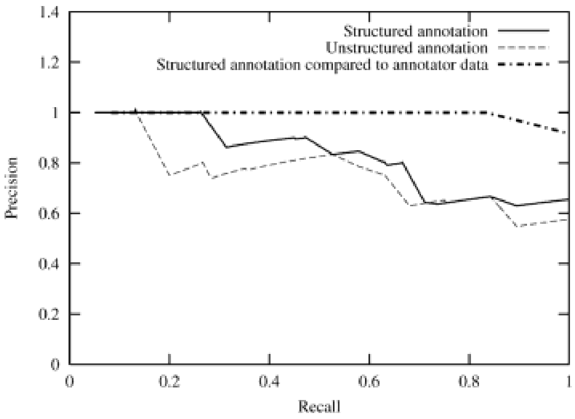
The images were presented in portions of four images to the annotator on a 1024*768 pixel 13.3 (=33.7 cm) TFT panel. The resolution per image was 256*256 pixels. The annotator (the first author) had the opportunity to scroll back and forth both in the annotation and in the image collection. The average time spent per image was about six minutes; 40 images were annotated per day. After the complete annotation was done, a debugging pass was performed. Here, drifts in annotation strategy as well as typographical and syntax errors were corrected. For this we verified the syntactical correctness, and, whether test queries on the annotation were confirmed by the similarity judgments of the annotator.

We then performed eight QBE queries, using the annotation scheme and the retrieval method described in the previous sections. The performance of these QBE queries was evaluated using relevance data which were collected for the experiments with the *Viper* CBR described in Squire et al., (1999). User data was collected for five users with and without computer vision experience. Each one performed the queries by hand, thus providing for each image a list of images relevant to the query. We kept all results for each user, thus storing the whole range of user behavior. This enabled testing the performance on relevance feedback, as shown in Müller et al., (2000b). In our present experiment, these relevance judgments were used to obtain precision-recall graphs of one-shot-queries on structured annotation.

For all queries, the structured annotation performs at least as well as the equivalent unstructured annotation (derived from the structured annotation by suppressing the structure). However, once again, it becomes clear that the problem of QBE for images is ill-posed: what is considered as relevant differs widely between the test subjects. With most query images both structured and unstructured annotation reached perfect performance for at least one test user. With some other images there is an advantage for the structured annotation, as shown in figure 4.

Both annotation methods performed very badly on test images that showed buildings as the only noticeable image item. We see as an explanation that both the annotator and the test users have no architectural background, so the relevance data were rather a product of the visual impression than of the semantics. However, in architecture and art, established classification methods exist (Greenberg, 1993).[3]

[3] It is tempting to conclude that the absence of domain knowledge in our test users would approach the user judgments to the results which are



```
% some words describing the setting
setting(books).
setting(library).
setting(inside).
% people looking at books,
% and choosing them
thing(library).
modifies(public,§library).
thingS(bookshelf).
actor(person,5).
action(read).
performs(§read,§person).
action(choose).
performs(§choose,§person).
thingS(book!2).% WordNet-sense 2 of "book"
isPerformedOn(§read,§book).
isPerformedOn(§choose,§book).
action(stand).
performs(§person,§stand).
```

FIG. 4. An annotation example, describing five people in a library standing next to bookshelves, reading and choosing books. The precision-recall graph compares the performance of structured and the corresponding unstructured annotation for this example. The base for the precision-recall data is previously collected user data. To compare how well the annotation reproduced the annotator's goal to express "people reading and/or choosing books," we also indicated how well the annotation performed compared with the annotator's relevance judgments.

The main use of adding such domain knowledge to the annotation process is to provide the annotator with a framework for consistent annotation. In most cases, this framework is given implicitly: it is established by the common culture.

We also experimented with the use of a thesaurus for improving the retrieval performance. We found that in our scenario, synonym sets and synonym disambiguation can be used in a beneficial way (volume-book instead of volume of liquid, for example). In using WordNet (Fellbaum, 1998), we experienced performance improvements when replacing each word of the annotation by the number of the WordNet synset to which it belongs. We would like to underline that also in this case, disambiguation has been done by hand in order to improve the query result. Trying to add WordNet hypernyms (generalizations) to each annotation item in a straightforward way degraded the query results. We identified as the principal reason for this: WordNet was built from a linguistic, but not from a visual perspective. For example, the word «Comic book» is visually very close to «book». Looking at a 256 × 256 pixel image, it is difficult to discern if a person is reading a comic book or a normal book. However, their WordNet hypernym hierarchies are very different: (**comic book**) → (magazine, mag) → (press, public press) → (print media) → (medium) → (means) → (instrumentality, instrumentation) → (artifact, artefact) → (object, physical object) → (entity, something), *whereas* (**book, volume**) → (product, production) → (creation) → (artifact, artefact) → (object, physical object) → (entity, something). We suggest adding hand-selected hypernyms where appropriate (policeman-man) leading to the creation of a visually as well as semantically inspired thesaurus for each collection. In addition to accounting for the specificities of the image annotation problem, this also accounts for the specificities of the collection. A visual thesaurus has been suggested in Picard (1995). However, what we suggest here is a thesaurus that includes both semantic and visual relations, thus expressing the relations that are seen by someone viewing an image while recognizing the semantics: for example, someone reading a folded newspaper can look similar to someone reading a book, someone reading an unfolded newspaper will look rather like someone reading a map. Looking for images of people reading a book we will prefer the former over the latter.

Ten days after finishing the annotation and the experiment, the annotator performed on the example queries by hand, comparing these relevance results to the results of

obtained by the CBR. This is not necessarily the case, as it is still largely a matter of personal choice *which* visual features lead the user to his relevance decision; however, using multilevel relevance feedback for learning feature weights we were able to obtain excellent results.

a query on his annotation. The results are shown in Fig. 4. We see that the annotation gave almost-perfect performance compared to the annotator's relevance judgment. Perfect agreement was achieved for the simple queries of our test query set. We concluded that our annotation can be used as user simulation for enabling a benchmark for image browsers.

The Benchmarking Process

Benchmarking a Bayesian CBR

We have implemented a benchmarking system that takes into account that interaction concepts of hierarchical browsers and stochastic browsers differ. In hierarchical browsers, the user needs to backtrack actively, if he or she reaches one leaf of the hierarchy. A stochastic browser will present suggestion after suggestion until the target image is found. Our benchmarking system uses pluggable components to adapt to the interaction strategy that is used by the system to be evaluated. It uses the Multimedia Retrieval Markup Language (MRML, described in Müller et al., (2000c) for the communication between the benchmarking system and the system to be tested, thus providing a common, easy-to-use access to the benchmark.

We applied our benchmarking system on a CBR that used `PicHunter`'s Bayesian retrieval method. Our `PicHunter`-like system used a color histogram distance and an image shape spectrum (Nastar, 1997) based distance measure. In the following, we will call this system `QuickHunter`.

`PicHunter` maintains for each image *I* the probability distribution that *I* is the target. It uses the user feedback to update this probability distribution, presenting at each step a set of images to the user. The set of images is presented to the user such that the expected duration of the search (expressed in the number of query steps) is minimized. Once shown to the user, images are discarded from the search.

For performing the benchmark we need three entities: The *benchmarking system* (abbreviated USim, as it contains the *user simulation*). The USim dispatches queries to the *benchmarked system* (`QuickHunter` in this case) and an *annotation-based query engine* (AQE, as described in previous sections), which provides the distance measure needed for the user simulation.

For measuring the performance of `QuickHunter` the USim performed a target test for a list of target images. For each, the USim performed a query by example on the AQE. The query result was pruned by hand so that it contained images related to the query only. Normalized, this ranked list served as d_{text}. Next the USim requested a (random) selection of 9 images from `QuickHunter`. The image of the selection with the smallest d_{text} was marked positive, the one with the biggest d_{text} was marked negative, and these two images were submitted as a query to `QuickHunter`. `QuickHunter` used this to calculate a new selection of 9 images, based on its d_{browser} distance measure. The process of giving feedback and requesting a new selection was repeated till the target image was found. On finding the target image, the search was re-initialized, and the process was repeated for the next target image.

TABLE 1. Benchmarking results for a PicHunter-like system, QuickHunter, for eight queries on TSR500.

Query	# images[a]	Remarks
One dollar	23	Perfect user agreement to the annotation
500 German marks	50	Many almost relevant images
Corner view of building	140	Medium user agreement, many almost-relevant images
Library	150	High user agreement visually inhomogeneous relevant set
Parliament	165	Same as above.
Lemons	241	Perfect user agreement to the annotation, small set of related images (3)
Harbor	255	Perfect agreement to annotation, but very small set of visually similar related images
Russian palace	315	Very bad user agreement with this annotation

[a] # images designates the number of images that had to be seen by the simulated user before finding the target.

For each target image, we counted the number of images shown to the user before the target was found by `QuickHunter`. As query images, we used the same images as for the evaluation of the annotation. The results are shown in Table 1. On average, `QuickHunter` needed to scan 170 images before the target was found, being more efficient than random search (250 images) by about 30%. In these experiments, the performances of `QuickHunter` varied by a factor of 10, depending on the «difficulty» of the target. As a consequence, we do not think that at the current state of research, the target testing performance of a system can be summarized by a single number (Müller et al., 2000a). We suggest giving both a detailed list containing the average performance of the image browser for each target image *and* the average performance over all target images.

Obviously, the combination of our image collection and our benchmarking method is hard for systems that do not try to adapt their user model during retrieval. A further difficulty is that in realistic collections, the number of images that are in any relationship with the target image is very small. This limits the possibilities of the USim to give useful feedback, as it does with real users.

However, in our opinion these are shortcomings of current browsing systems, problems that have to be addressed in future research. For the time being, to measure the progress of image browsers towards the use for semantic queries, we propose a migration path towards semantic benchmarking: use a superposition of a semantics-derived and a visual (low-level feature) distance measure for user simulation. The weight with which both distance measures are superimposed could then be used to describe the degree of difficulty of the benchmark.

Conclusion

We propose an automatic semantic-based benchmark for image browsing systems. The advantage of such a benchmark is that it is memoryless, and not influenced by imponderables like the previous experience of test users.

We based our automatic benchmark on structured annotation that is augmented using a thesaurus. For this benchmark we developed an efficient query method for semantic networks that performs ranked similarity queries using inverted files followed by a stage where more elaborate matching is performed.

We see the use of this work as two-fold: first, the proposed structure allows studying the interweaving between annotation and still image features, especially still image segments. The retrieval method proposed allows for inclusion of derived visual features with and without annotation. Secondly, the benchmark devised in this article constitutes a tool, both for the development and for the evaluation of image browsing systems. We hope it will stimulate more research for intelligent systems that learn the *semantics* of a query during the querying process.

References

Cox, I. J., Miller, M. L., Omohundro, S. M., & Yianilos, P. N. (1996). Target testing and the PicHunter Bayesian multimedia retrieval system. In Advances in Digital Libraries (ADL '96), pages 66–75, Library of Congress, Washington, D. C.

Fellbaum, C. (Ed.). (1998). WordNet—An Electronic Lexical Database. MIT Press, Cambridge.

Greenberg, J. (1993). Intellectual control of visual archives: A comparison between the art & architecture thesaurus and the Library of Congress thesaurus for graphic materials. Cataloging and Classification Quarterly, 16(1), 85–117.

Huijsmans, D. P. & Smeulders, A. W. M. (Eds.). (1999). Third International Conference On Visual Information Systems (VISUAL '99), number 1614 in Lecture Notes in Computer Science, Amsterdam, The Netherlands. Springer-Verlag.

Müller, H., Müller, W., Squire, D. M., Marchand-Maillet, S., & Pun, T. (2001). Performance evaluation in content-based image retrieval: Overview and proposals. Pattern Recognition Letters, 22(5), 593–601.

Müller, H., Müller, W., Squire, D. M., Marchand-Maillet, S., & Pun, T. (2000b). Strategies for positive and negative relevance feedback in image retrieval. In Proceedings of the 15th International Conference on Pattern Recognition (ICPR 2000), Barcelona, Spain. IEEE.

Müller, W., Müller, H., Marchand-Maillet, S., Pun, T., Squire, D. M., Pečenović, Z., Giess, C., & de Vries, A. P. (2000c). MRML: A Communication Protocol for Interoperability and Benchmarking of Multimedia Information Retrieval Systems. In Smith, J. R., Le, C., & Panchanatan, S., editors, Internet Multimedia Management Systems, volume 4210 of SPIE Proceedings, Boston, Massachusetts, USA. (SPIE Information Technologies 2000).

Müller, W., Pečenović, Z., de Vries, A. P., Squire, D. M., Müller, H., & Pun, T. (1999a). MRML: Towards an extensible standard for multimedia querying and benchmarking—Draft proposal. Technical Report 99.04, Computer Vision Group, Computing Centre, University of Geneva, rue General Dufour, 24, CH-1211 Geneve, Switzerland.

Müller, W., Squire, D. M., Müller, H., & Pun, T. (1999b). Hunting moving targets: an extension to Bayesian methods in multimedia databases. Technical Report 99.03, Computer Vision Group, Computing Centre, University of Geneva, rue General Dufour, 24, CH-1211 Geneve, Switzerland.

Nastar, C. (1997). The image shape spectrum for image retrieval. Technical Report RR-3206, INRIA, Rocquencourt, France.

Ounis, I. (1999). A Flexible Weighting Scheme for Multimedia Documents. In Proceedings of the 10th DEXA International Conference on Database and EXpert Systems Applications, pages 392–405, Florence, Italy.

Ounis, I. & Pasca, M. (1998). Modeling, indexing and retrieving images using conceptual graphs. In Proceedings of the 9th DEXA International Conference on Database and EXpert Systems Applications, pages 226–239, Vienna, Austria.

Papathomas, T. V., Conway, T. E., Cox, I. J., Ghosn, J., Miller, M. L., Minka, T. P., & Yianilos, P. N. (1998). Psychophysical studies of the performance of an image database retrieval system. In Rogowitz, B. E. and Pappas, T. N., (Eds.), Human Vision and Electronic Imaging III, volume 3299 of SPIE Proceedings, pages 591–602.

Picard, R. W. (1995). Toward a visual thesaurus. Technical Report 358, MIT Media Laboratory Perceptual Computing Section, 20 Ames St., Cambridge MA 02139.

Smith, J. R. & Chang, S.-F. (1996). Tools and techniques for color image retrieval. In Sethi, I. K. and Jain, R. C., (Eds.), Storage & Retrieval for Image and Video Databases IV, volume 2670 of IS&T/SPIE Proceedings, pages 426–437, San Jose, CA, USA.

Squire, D. M., Müller, W., & Müller, H. (1999). Relevance feedback and term weighting schemes for content-based image retrieval. In Huijsmans and Smeulders, 1999, pages 549–556.

TREC (1999). Text REtrieval Conference (TREC). http://trec.nist.gov/.

Vendrig, J., Worring, M., & Smeulders, A. W. M. (1999). Filter image browsing: Exploiting interaction in image retrieval. In Huijsmans and Smeulders, 1999, pages 147–154.

Voorhees, E. (1999). Natural language processing and information retrieval. In Pazienza, M. T., (Ed.), Information Extraction: Towards Scalable, Adaptable Systems, volume 1714, pages 32–48. Springer Verlag.