Tangible Image Query

Krešimir Matković[1], Thomas Psik[2], Ina Wagner[2], and Werner Purgathofer[3]

[1] VRVis Research Center in Vienna, Austria,
http://www.vrvis.at, Matkovic@VRVis.at
[2] Institute for Design and Assessment of Technology,
Vienna University of Technology, Austria,
http://www.media.tuwien.ac.at, {psik, iwagner}@pop.tuwien.ac.at.at
[3] Institute of Computer Graphics and Algorithms,
Vienna University of Technology, Austria,
http://www.cg.tuwien.ac.at, wp@cg.tuwien.ac.at

Abstract. This paper introduces a tangible user interface for browsing and retrieving images from an image database. The basis for the query to the image database is a color layout sketch, which is used by the underlying query algorithm to find the best matches. The users are provided with colored cubes of various sizes and colors. The users can place and arrange the colored cubes on a small table to create a color layout sketch. Multiple users can use this interface to collaborate in an image query. To evaluate the benefits of the interface, it is compared to a traditional GUI application in which the users use a mouse to paint a color layout sketch. The tangible interface appears to be easier to use and better accepted by people who belive they are unable to draw or paint or who do not want to use computer.

1 Introduction

The explosion of digital technology in the last decade led to an enormous amount of digital images. Conventional ways of data retrieval became just insufficient for large amounts of visual material. Popular thumbnail views are useless, if we have thousands or tens of thousands of images. Another approach, key wording, simple does not work for most of us. It is easy to keyword a few images, but it is illusory to expect that average users will keyword their whole collection of images. Eakins and Graham [1] claim that some professional agencies need up to 40 minutes to keyword a single image. It is clear that common users confronted with hundreds and thousands of images cannot do such precise key wording. Content based image retrieval, which has been a subject of extensive research in the last decade, tries to offer a solution for retrieving images from large databases.

The original and still often used idea is the query by example method. This means that the user supplies an image, and the system tries to find similar images. In this case the central problem is the definition of similarity. As humans themselves can not always agree on what is similar and what is not (or what is more similar) the results of image retrieval are often unexpected and sometimes disappointing. Figure 1 shows an example where such a system was used to search for images similar to the bird image. If the user understands that the system tries to find images with similar color layout,

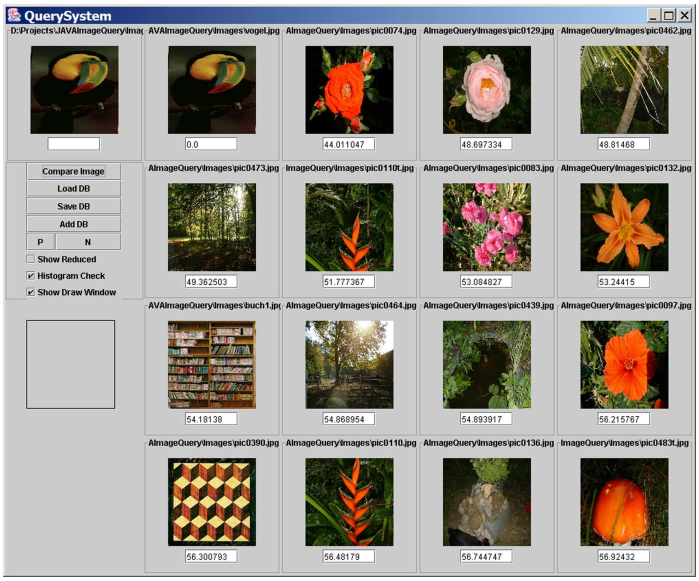

Fig. 1. Query by Example can be disappointing if the user does not understand the underlying algorithm. Here the system searches for a similar color-layout, and not for birds.

and not content (bird in this case), results are much more satisfactory. On the other hand if the user expects birds she/he might be really disappointed.

The next step in image retrieval was not to search only for overall similarity, but rather to find images containing a specific pattern. A company logo is a good example. Imagine a company searching for images containing their logo. The logo can be anywhere in the image, it can be taken under various lighting conditions, it can be distorted due to the perspective projection and so on. Clearly this is not a trivial task. Furthermore, if one tries to find all images containing a bird, for example, the whole search becomes practically impossible.

There are numerous systems capable of various kinds of image queries available. IBM's QBIC System [2] was one of the first systems, and it can be tested online at [3, 4]. The VIR Image engine [5] from Virage, Inc. and the Photobook Project [6] developed in the MIT Media Lab are two also well known examples. The work of Jacobs at al. [7] is especially well known in the computer graphics community. All of these as well as [8–10], represent the query by example approach. There are systems like Blobworld [11, 12] or ICONA [13, 14] which represent another group of systems, they go beyond simple query by example, and try to find similar shapes, textures, and other features.

Some systems offer a possibility for the user to sketch the query image. The basic idea is that a user might remember how the image looked like (but cannot remember the image's name), so the user sketches the image, and the system finds matching images from the database. Another possible scenario of use comes from the designers' and architects' perspective. In the concept design phase of a project it is common practice to browse through image collections in order to be inspired, to see some unexpected connection between images. Visual image query can be used for such a browsing. The

drawback of the method described above (see Figure 1) suddenly becomes an advantage. Asking for a parrot, and getting a flower can be either: frustrating or inspiring, depending on the user and the context.

This work is based on such a system, and a new kind of user interface for sketching images is introduced. Instead of using a mouse to draw, users are provided with small cubes of various sizes and colors, and they try to sketch an image using the color cubes. Cubes are placed on a semitransparent glass surface. Besides the cubes, users may use any colored objects. This kind of "sketching" using currently available artifacts is particularly common among designers and architects. We implemented the method, built a prototype and tested it with users. Finally we compared the results with conventional sketching using a mouse.

2 Underlying algorithm

Although the underlying system is arbitrary, and the method can be combined with any query by example method, we needed one method for our implementation. The system is based on the visual image query by Matkovic et al. [15]. The underlying algorithm will be briefly described. Just like most of the image query methods, the method uses descriptors calculated for each image. These descriptors are created in the database during a preprocessing phase. When the user performs a query, a descriptor is created for the query image and compared to the stored descriptors. Various query systems differ in the way how descriptors (sometimes called signatures) are created. The Visual Image Query (the system he have used) calculates descriptors using 1000 quasi-randomly distributed rectangles of various sizes in the image. The rectangles partly overlap. The sizes of the rectangle are chosen according to the contrast sensitivity function of the human eye. Figure 2 illustrates the distribution at the first 100, 250, 500, and 1000 rectangles. For each rectangle the average color is computed, and all 1000 Luv color triples are stored in the signature. The signature contains only color information for each rectangle, and the system can not distinguish if, e.g., an orange spot in the middle is a flower or a fish. The only information known is that there is an orange spot in the middle. The exact shape of the spot is also not known. It is sampled using the rectangles, and can never be precisely reconstructed. The comparison of two descriptors is done in the Luv color space, i.e. for all 1000 triples the luv-difference is computed and a weighted sum, according to the contrast sensitivity function, of these differences is taken as distance function of the two images.

This method was selected since it is particularly convenient for the comparison of user sketches. The sketch is not precise, and actually, only the color layout matters. However, in order to make it suitable for the new interface, and in order to compare it with conventional input, the original algorithm had to be changed slightly.

2.1 Changes to the original algorithm

In the original algorithm the descriptor consists of 1000 Luv triples. Comparing two descriptors means computing the Luv difference for 1000 triples. In order to speed up the process, the algorithm was modified slightly. 1000 rectangles are placed in the image, and the average color of each rectangle is computed. This average color is then rounded to the 64 color set. Now, the descriptor consists of 1000 indexes (in the 64 color set) instead of 1000 Luv triples. The difference between two descriptors can be computed faster using a matrix of predefined differences between all 64 available colors.

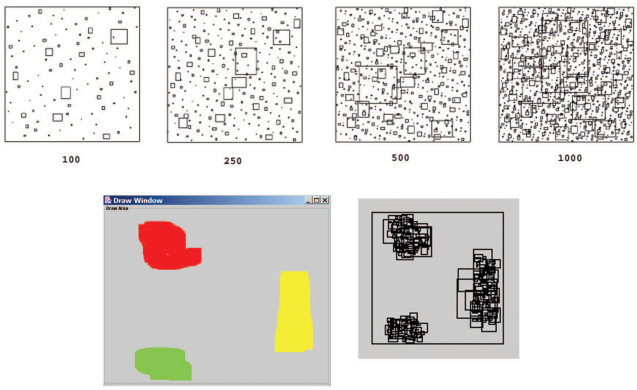

Fig. 2. Top: Rectangle distribution for the first 100, 250, 500 and 1000 rectangles in the algorithm we have used.Bottom: The user draws three separate areas, and only the corresponding rectangles are used for this query.

In the original algorithm either the whole image or one selected area was compared. This had to be changed to allow multiple areas. Only these parts of the image where the user sketched something will be used in the comparison. In this way the user does not need to sketch the background, but only significant places she/he remembers. Furthermore, the query starts automatically when the user does not change the sketch for a second, and results are displayed. Figure 2 illustrates an example of a simple sketch and the subset of rectangles used in this case. Of course, the support for the new interface had to be added as well.

2.2 Sketching the query image

Tests with the original system using conventional mouse input, showed that there are two groups of users. The first group of users, forming a majority, are the users who claim they cannot draw (or paint, or sketch). It was not easy to encourage them to try the system. They were just saying "I can not draw". Although we explained that they do not need to draw an exact reproduction, but just a red spot here, and a blue spot there... just a color layout sketch, it was still not easy to get sketches from them.

The second group of users were users who can draw. The problem with them was that they were not satisfied with the sketch, they wanted to have it perfect.

Some systems are offering tools for drawing circles, rectangles, and other primitives. If such a system is used for color layout search results are even more disappointing. Imagine a user drawing a yellow circle, and the system responding with flower images, or even yellow triangles. Of course, the system was not recognizing the shape, but only the color. This is misleading for the users in most cases.

It was clear that conventional sketching is a good solution only for a very limited number of users. Another kind of interface is needed, an interface that is very suitable for sketching, but which is not suitable for drawing. In this way, users who cannot draw will not be disappointed with their drawing results. It is impossible to draw with that interface anyhow, and for the same reason the users who can draw will not try

to draw perfectly. Such an interface is introduced in this paper, and this is the main contribution.

3 New sketching interface

The whole setup consists of a table with a semi-transparent white glass plate. There is a set of color cubes, and the users can arrange them on the table in order to make a sketch. A simple web cam is mounted under the plate, which is used to retrieve the color layout sketch. This sketch image is then used as a query image. Figure 3 shows a part of the setup with the table used for sketching. It was common practice during our experiments that users "draw" together. They stood around the table, and instead of the others instructing one user what to do (which was common with the mouse), the group could draw together. The collaboration is another important quality of the cubes interface. Furthermore, not only the cubes can be used to sketch. As soon as a bowl of fruits was placed next to the table, some users used oranges and apples as sketch tools.

Fig. 3. Students experimenting with the new interface.

4 Vision based color sketch

The Crayon project [16] provides a good overview of the current state of vision based interaction. In the project the researchers use a camera for hand tracking and explored the field of color calibration and machine learning. Our approach is related to their work in the respect that we also extract color information from a live video stream.

Various problems that are related to color vision had to be faced. First tests showed that for certain colors (especially cyan and gray) that were desirable, no stable calibration was possible. This is because web cams provide compressed video information and use optical sensors that are optimized to capture images of faces. The main usage of this kind of camera are video meetings, so the red part of the visual spectrum is covered quite well, but blue and contrast are not of high concern.

4.1 Hardware setup

To reduce the problems that come with computer vision, like changing ambient light and dynamic in- camera adaption, a setup where these external interferences are reduced was created. The camera was mounted underneath a semi transparent surface, on which the colored cubes were placed. Also a light source was installed underneath this surface to ensure proper lighting conditions. The setup was surrounded by a non transparent casing leaving only the surface visible to the users and exposed to the ambient light in the room. It was possible to achieve good results with a static calibration of the color detector with this setup. The output of the query was displayed by using a projector to create a large screen right in front of the image sketching surface.

The need for such a special hardware setup might be considered to be a drawback of the system. Not everyone has the possibility to allow extra space in the office for such an installation. In such a case a simplified system consisting of a web cam pointing down on the desktop (the real desktop), and a set of colorful cubes, game stones, pieces of paper, or similar things can be used to test the system. Of course, the system is more single user oriented in such a reduced setup, but it suffices for test purposes. Furthermore, the use of flatbed scanners for this purpose was briefly exploited with advantages like better color and contrast but also drawbacks like increased response times and reduced interactivity.

4.2 The steps of the color sketch retrieval

Two approaches were implemented to create our test setups. In the first implementation the color segmentation was applied at the vision part of the system. First an image is grabbed from the web cam, then a color segmentation is performed and finally a color indexed image is sent to the search engine. The color segmentation was implemented using the HSV (hue, saturation, value) color space. For each color (white, yellow, orange, red, green, blue, magenta, black) ranges for the HSV values are specified. Using a simple filter, regions in the grabbed image that have color values within these ranges are copied to a color index image. The color index range is from 1 to 8. Zero is being used to indicate that none of the colors was detected. This indexed color image is then sent over the network to the search algorithm.

In the second implementation background subtraction was used to filter out the parts of the video stream that have been changed or added by the users. This approach sends a full color image (with reduced size) and an alpha channel (specifying the regions that are not background) to the search engine.

A "change" parameter is extracted from the live stream as well, measuring how much the image has changed between two updates. A high value indicates that the users are currently changing the sketch or just moving the hands within the observed area (for example to point out certain regions and compare them with the results). During this period of vivid interaction no update is sent to the search algorithm, not

even the color segmentation or background subtraction is evaluated. Such intermediate results would confuse the users and also distract their concentration from the task of creating or changing a sketch. When the "change" parameter drops below a certain value the image segmentation is activated. If the difference between the resulting sketch and the previous query to the search algorithm is above a certain value (indicating that the vivid change in the video stream was not just moving the hand but also moving some objects), the new sketch is sent to the search algorithm. This makes it possible to create fast update rates, as no unnecessary video images and queries are evaluated.

Both image segmentation approaches have their advantages. The color segmentation provides better results in respect of removing the background and not used areas. Because the background subtraction algorithm dynamically updates the reference image it is more stable to ambient light changes. Also the background subtraction allows the use of more than 8 colors, because the colors are not mapped to one of the indexed colors of the cubes. At the same time the network traffic increases as more data has to be sent to the search engine.

Selection of colors for the tangible interface The original implementation using the mouse as an interface allowed the users to create a sketch with about 50 different colors. Users memorizes only main colors mostly based on the hue value. They tend to use only a basic set of colors when they try to reproduce an image. Therefore to help the users to focus on a basic color sketch, only basic colors where provided in form of colored cubes for the tangible interface. White and black as representatives for the grey spectrum and red, green, blue as the basic colors. As yellow, orange and magenta are also well memorized colors those were provided, too. Early tests with a web cam showed that cyan as a mixture of green and blue is badly captured by web cams. Therefore no cyan cubes were provided for the tests. Feedback from the users proved that cyan is not an important color.

5 Comparing mouse and tangible interface

As discussed in [17] a user interface can be evaluated with the terms: degree of indirection, degree of integration, degree of compatibility. Although the original publication focuses on widgets, it can also be adopted for tangible user interfaces. The object that the interface operates on can be interpreted in two ways. On the one hand the users manipulate the color layout sketch, on the other hand they do that because they want to change the results of the color layout query.

The degree of indirection is a measure of the spatial and temporal offsets generated by an interface. The spatial offset is the distance between the input part of the interface and the object it operates on. The temporal offset is the time difference between the physical action on the interface and the response of the object. The temporal offset is quite the same for both interfaces, as the sketching of the color layout is performed in real time with both interfaces, without any time delay. And after a specific time without manipulation both interfaces send the created sketch to the search algorithm. The spatial offset is slightly better with the mouse interface as the drawing area and the display of the results are on the same screen and the tangible interface needs two separate areas, one to sketch the color layout and one to present the results.

The degree of compatibility measures the similarity between the physical actions of the users on the interface and the response of the object. The tangible user interface

provides a higher degree of compatibility as the users directly manipulate the color layout sketch with the colored cubes. The interface is a very direct approach without abstract mapping between input and effect on the query. With the mouse interface the users have to draw by selecting a color from the palette and then move the mouse to create a colored area in the sketching window.

The degree of integration measures the ratio between the degrees of freedom (DOF) provided by the logical part of the instrument and the DOFs captured by the input device. The term degree of integration was introduced in integral tasks [18]. The degree of freedom can be evaluated in two dimensions: the color dimension and the layout (2D) dimension. The mouse interface provides only a 2D interface. Therefore an indirect color selection method has to be incorporated. The tangible interface in our current setup allows direct access to all three dimensions (color and 2D), but as one of our test users stated, the cubes can also be stacked to create a three dimensional structure. So the tangible interface has four dimensions that can be operated on. These do not match with the needed three dimensions, but can be resolved if colored objects are used that cannot be stacked, like half-spheres instead of cubes.

6 Results

We have tested the system with a large number of users. The application was presented at various workshops, including a "Beginner's day" at the university and a workshop that was part of the review of the ATELIER project. Some of the workshops were publicly accessible, therefore different types of users tested the system. The users had different drawing and computer usage skills. The general feedback from the testers was very positive. The tangible interface is very attractive and easy to use.

In addition he have interviewed selected special users, that work with pictures in their profession. A collection of approximately one thousand images was presented to the users. They observed a slideshow, and they were asked to remember a few images that had an impact on them. Afterwards they tried to draw a sketch in order to retrieve the memorized images. First they made a sketch with the mouse, and then using the new color cube interface. Results and impressions of users were compared at the end.

Figure 4 shows a sketch of a sitting person, and result images (she was looking for the second best guess by the system). Most of the users tried to "draw" the picture with the mouse, and the tangible interface helped them to understand that a sketch is better for the search than a "redraw" of the image they searched for. The results that were presented by the search algorithm often did not fit their expectations when drawing with the mouse (and trying to draw structures). The color they found was missing were mostly brown and grey, but this could correlate with the picture selection. Investigating the multi user aspect of the tangible user interface was also interesting, as some of the users complained about it, when other destroyed their work by changing the sketch without asking.

The general response was very good, and most of the users liked the tangible interface better than conventional one. As we had some test users with visual arts background, we noted that they were very pleased with the surprising component of the tool. E.g., a user searched for a sun-set that was instantly within the top 15, but mixed with images of red flowers and a firework. These results were far from disappointing, and the flowers and firework images fitted well in the users expectations.

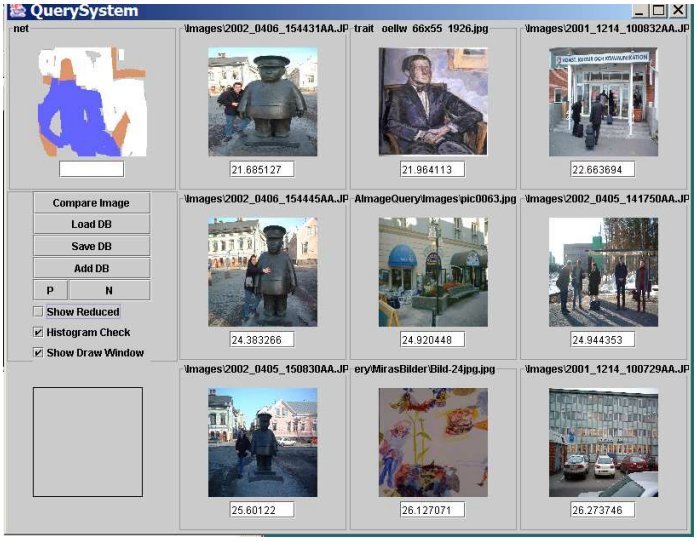

Fig. 4. Sketch done by a painter looking for the portrait of the sitting person.

7 Conclusion

As many available examples prove the color layout search is an interesting approach to image query. Our work presents a new tangible user interface that allows creating color layout sketches in an easy and straight forward manner. Rather than improving the query algorithm itself, we tried to find a new interface which suits the existing algorithms better. The algorithm needs a certain level of abstraction, which is often hard to achieve using a traditional mouse-interface. The new color cube interface makes it impossible to draw precisely, and therefore helps the users achieve the needed level of abstraction.

Still many of the problems of the underlying methods persist. Tangible user interfaces enrich the possibility of collaboration and multi user input, with all the problems that come with it. For example there is no method of helping the users with synchronization, as all users that use the interface, actually shared this interface physically. They have to sort out conflicts between them without (computational) help, for example: someone adding his ideas to the sketch without asking. The method of color layout image retrieval also has its flaws. Most of the users cannot clearly identify the distinction between shape and color layout. A good example is the search for a sunset. A red shape placed in the middle of the image is a good approach, but images where the sun is not close to the center will not be found, even if it is a picture of a sunset, and images of a red flower in the center of the image will be found instead.

We observed that the use of a tangible user interface helps the users to create color layouts rather than shapes. More over the interface can be used in a more vivid way. It allows direct access to the sketch rather than the indirect method of using a mouse.

The color cubes interface fits very well with the underlying visual image query, and helps the users to cope with the limitations of the query algorithm. In this way the usability of the whole system is significantly enhanced.

8 Future work

We want to integrate this interface into a framework, where designers are adding images to a repository. These pictures are indexed (with words) and therefore we will be able to do a filtering based on these keywords, also. This will lead to an image query system that combines the unsharp search based on color layout, as described in this article, and image search by keyword. In combining these two approaches we hope to encourage the users of the framework to make use of this interface even more. We want to test whether the combination of indexing by words and a color based image search will result in a better interface or not. The surprise element of results will surely decrease (in case the indexing is done properly), but also the results could fit better to the expectations of the users.

Improvements on the color layout search engine will also be investigated. As we will have access to a repository with over 10.000 images, we can than test the scalability of the algorithm and probably introduce new aspects in respect of clustering the database and improving the response to the query.

The vision system as described in this paper was realized using a consumer web cam. If a high quality camera can be used, surely the detection of the color sketch would improve as the color dynamic will increase.

9 Acknowledgments

The authors would like to thank our co-researchers from the Atelier Project, in particular Andreas Rumpfhuber. This work was partly sponsored by the European Commission - IST programme - Future and Emerging Technologies - Proactive Initiative - The Disappearing Computer II - through the ATELIER project (EU IST-2001-33064). Parts of this work were carried out at the VRVis Research Center in Vienna (`http://www.VRVis.at/`), Austria, which is funded by an Austrian governmental research program called K plus.

References

1. Eakins, J.P., Graham, M.E.: Content-based image retrieval, a report to the jisc technology applications programme 1999. (Technical report)
2. Petkovic, D., Niblack, W., Flickner, M., Steele, D., Lee, D., Yin, J., Hafner, J., Tung, F., Treat, H., Dow, R., Gee, M., Vo, M., Vo, P., Holt, B., Hethorn, J., Weiss, K., Elliott, P., Bird, C.: Recent applications of ibm's query by image content (qbic). In: Proceedings of the 1996 ACM symposium on Applied Computing, ACM Press (1996) 2–6
3. IRS: (Image Retrieval Service (IRS) of the EVlib, http://visinfo.zib.de/irs)
4. QBIC: (The State Hermitage Museum, St. Petersburg, Russia, QBIC Color and Layout Search, http://www.hermitagemuseum.org/fcgi-bin/db2www/qbicsearch.mac/qbic?sellang=english)
5. Gupta, A.: The virage image search engine: an open framework for image management. In: Storage and Retrieval for Image and Video Databases IV, SPIE proceedings series. Volume 2670. (1996) 76–87
6. Pentland, A., Picard, R., Sclaroff, S.: Photobook: Content-based manipulation of image databases. In: SPIE Storage and Retrieval for Image and Video Databases II, number 2185, Feb. 1994, San Jose, CA. (1994)

7. Jacobs, C.E., Finkelstein, A., Salesin, D.H.: Fast multiresolution image querying. In: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, ACM Press (1995) 277–286
8. Faloutsos, C., Barber, R., Flickner, M., Hafner, J., Niblack, W., Petkovic, D., Equitz, W.: Efficient and effective querying by image content. Journal of Intelligent Information Systems **3** (1994) 231–262
9. Kelly, P.M., Cannon, M.: Query by image example: The candid approach, los alamos national laboratory white paper (1995)
10. Vailaya, A., Zhong, Y., Jain, A.: A hierarchical system for efficient image retrieval,. In: Procedeengs of International Conference on Pattern Recognition (August 1996). (1996)
11. Belongie, S., Carson, C., Greenspan, H., Malik, J.: Color- and texture-based image segmentation using EM and its application to content-based image retrieval. In: Proceedings of the Sixth International Conference on Computer Vision. (1998)
12. Carson, C., Thomas, M., Belongie, S., Hellerstein, J.M., Malik, J.: Blobworld: A system for region-based image indexing and retrieval. In: Third International Conference on Visual Information Systems, Springer (1999)
13. Boujemaa, N., Fauqueur, J., Ferecatu, M., Fleuret, F., Gouet, V., Saux, B.L., Sahbi, H.: Ikona for interactive specific and generic image retrieval. In: Proceedings of International workshop on Multimedia Content-Based Indexing and Retrieval (MMCBIR'2001), Rocquencourt, France. (2001)
14. Fauqueur, J., Boujemaa, N.: Logical query composition from local visual feature thesaurus. In: Proceedings of Third International Workshop on Content-Based Multimedia Indexing (CBMI'03). (2003)
15. Matkovic, K., Neumann, L., Siglaer, J., Kompast, M., Purgathofer, W.: Visual image query. In: Proceedings of Smart Graphics 2002. (2002)
16. Fails, J.A., Olsen, D.R.: A design tool for camera-based interaction. In: Proceedings of the ACM CHI 2003 Conference on Human Factors in Computing Systems, Association for Computer Machinery (2003) 449–456
17. Beaudouin-Lafon, M.: Instrumental interaction: An interaction model for designing post- wimp user interfaces. In: Proceedings of the ACM CHI 2000 Conference on Human Factors in Computing Systems, Association for Computer Machinery (2000) 446–453
18. Jacob, I., Oliver, J.: Evaluation of techniques for specifying 3d rotations with a 2d input device. In: Proceedings of HCI'95 Conference, People and Computers X. (1995) 63–76