Querying by Photographs: A VR Metaphor for Image Retrieval

Jürgen Assfalg and Pietro Pala
University of Florence, Italy

Interaction paradigms based on 3D interfaces and virtual reality offer new possibilities to overcome the limitations of query by example. We present a system that lets users navigate a 3D world where they can take photographs to query a database of images by content. Furthermore, users can interactively customize the virtual world by adding objects to the scene and editing object properties such as colors and textures.

The emergence of multimedia technology and the possibility of sharing and distributing image data through large-bandwidth computer networks have contributed to an increase of visual data in the global information exchange. Significant advances have been made in the development of efficient compression techniques, but techniques that enable efficient retrieval by content of visual data remain an active research topic.

Recently, researchers have developed new tools and interaction paradigms to search for visual information by referring directly to its content. Visual elements such as color, texture, shape, structure, and spatial relationships serve as clues for retrieving images with similar content.

The most successful and commonly employed querying interfaces rely on query-by-example paradigms. These paradigms require users to sketch, either from scratch or using prototype images, the content of the image they're looking for. Although it's easy to create simple examples, producing significant examples of complex scenes remains a difficult task.

The photographer's metaphor presented in this article facilitates authoring complex examples. Actually, the user takes a photograph of a (customizable) 3D environment rendered by the system. Since the system renders the environment, the paradigm's effectiveness doesn't rely on the user's painting abilities.

Background

The interaction paradigm of querying by 2D visual examples suits content-based retrieval well. This paradigm requires users to provide a prototype image as a reference example to express perceptual aspects of low- and intermediate-level features of visual content.[1,2] Several types of querying by 2D examples have been proposed so far, including

- *Iconic querying*. Usually employed for retrieval by spatial relationships. Icons representing objects and concepts are placed on a blackboard to reproduce the objects and concepts and their arrangement within the image.
- *Querying by painting*. Usually employed for retrieval based on chromatic properties of image regions. The user sketches color patches and reproduces salient color regions and their spatial arrangement.
- *Querying by sketch*. Commonly used to retrieve images that contain objects with shapes similar to the example. Sketches can be manually authored or traced from an image, following the objects' contours.
- *Querying by image*. Especially suitable for queries addressing global color or texture similarity. A sample image, taken either from the answer to a previous query or from a sample image set, serves as a prototype to retrieve similar images.

Although widely employed in current systems for content-based retrieval, querying by 2D-examples has several drawbacks, as summarized in Table 1. First, users find it difficult to create suitable copies of the images they want to retrieve by scratch (as needed in querying by painting or querying by sketch). This usually occurs because most users lack visual memory, painting and sketching skills, and the ability to detect salient features that characterize the visual content. As a result, users create examples according to the querying-by-image approach. Existing images, possibly retrieved through a previous query or selected from a random subset of the database, serve as examples. In some cases, the example barely represents the image the user wants to retrieve, and it's difficult to modify the example to adjust it to the desired content. The user can only select the whole image as it is (thus referring to its global properties) or a part of the image that best represents the queried content.

1070-986X/00/$10.00 © 2000 IEEE

Iconic querying represents a compromise between the complexity of drawing examples as in querying by sketch and the inadequacy to express content as in querying by image. Iconic querying mostly handles the representation of spatial relationships. Predefined icons representing object types are arranged in the appropriate positions to reproduce the searched images' spatial arrangement. Nevertheless, iconic querying has several limitations.

First, the number of icons can be extremely large, thus causing problems of organizing the objects and concepts they represent. Besides, editing the icons' properties is difficult, if not impossible. Finally, when a user has to represent a 2D view of a 3D scene, 2D icons aren't effective—even for representing spatial relationships.[3] A possible solution calls for overlapping sketches or icons to represent the arrangement of objects in the 3D scene. However, this approach's effectiveness is limited to simple scenes.

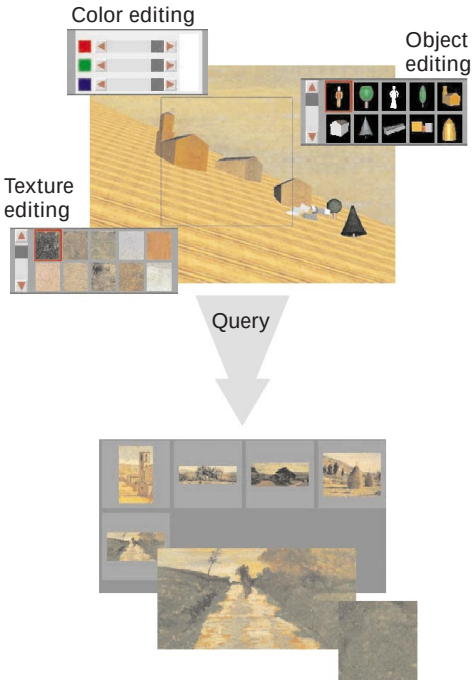
Interactive 3D iconic interfaces can solve most of the previously mentioned problems. Using 3D interfaces enhances the realism of the example and agrees with how humans perceive 3D scenes. As demonstrated by research in experimental and cognitive psychology,[4] the mental process of humans simulates physical world processes. Humans regard computer-generated line drawings representing 3D objects as 3D structures, not as image features. Three-dimensional interfaces let users navigate in a 3D scene. They can choose the most appropriate location and orientation for viewing the scene. Furthermore, 3D interactive graphics supports extended editing capabilities such as the possibility to dynamically customize the virtual environment both in terms of object color, texture, and shape features and in terms of their 3D spatial relationships.

Some researchers have addressed accessing image databases through a 3D interface.[3-6] In all these systems, 3D interfaces express queries only in terms of spatial relationships between objects. Hildebrandt and Tang[5] described retrieving 3D computer-aided design (CAD) models. They extended a 2D string description to represent and query 3D objects. Del Bimbo, Campanai, and Nesi[3] addressed retrieving 2D images as 3D real scenes. They introduced a 3D graphics interface with 3D icons to support querying and retrieval based on 3D spatial relationships.

Other systems that retrieve images or other kinds of artwork (such as sculptures, pottery, and so on), have used 3D mainly for visualization purposes. In other work[7-9] results of a user query were arranged in a 3D environment to enable the fruition of artwork in the same way as the user is accustomed to in real museums. Both approaches provide a dynamic creation of the scene, depending on the results of the query. In one system,[8] an advanced trigger mechanism allows the scene on the client side to track changes in the database. However, 3D-based interaction is limited to navigation in the environment and manipulation of the objects it contains, while query specification is based on Structured Query Language (SQL).

Table 1. Drawbacks of query-by-example paradigms.

Drawback	Image	Painting	Sketch	Iconic
Doesn't allow content editing	×			×
Examples don't fit user's requirements	×			
Requires too many images or icons	×			×
Requires painting or sketching abilities		×	×	
Doesn't give visual feedback				×
Requires visual memory		×	×	×



Images courtesy of De Luca Editore

Figure 1. Users can customize a scene using different colors, objects, and/or textures that can be accessed through different palettes. The selected portion of the framed scene (the photograph) is used to query the database. Retrieved images are then presented to the user for browsing.

The query metaphor

The querying paradigm proposed in this article aims to lessen the requirements for painting and/or sketching abilities. At the same time, it preserves the capability of editing the content of a query. The query paradigm relies on rendering 3D scenes that the user can interactively customize. A 3D graphics engine automatically projects the 3D virtual environment onto a 2D screen view (see Figure 1).

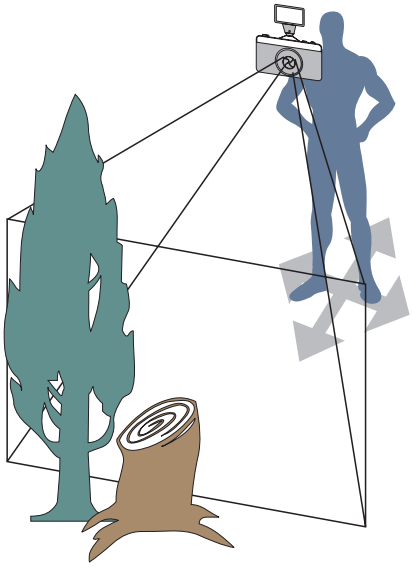


Figure 2. The user navigates the environment, continuously changing the viewpoint of the virtual camera. Running a query is as simple as taking a photograph of the scene.

The 3D interactive interface supports user navigation in the 3D environment. This allows for a nearly unlimited set of viewpoints from which the user can take some photographs (see Figure 2).

The 2D query images thus produced are passed to the retrieval engine and used as samples for content-based access to an artwork database. We call this querying paradigm the photographer's metaphor. Indeed, resorting to a metaphor helps explain the system to inexperienced users by referring to concepts of a new, unknown domain (the target domain) in terms of concepts borrowed from a well-known one (the source domain).[10,11] Therefore, the photographer metaphor makes image retrieval as easy as taking photographs.

By projecting the 3D scene into a 2D query for retrieval, depth information is lost. Therefore, the retrieval engine relies only on the information contained in 2D images, disregarding any information on the geometrical 3D structure of and spatial relationships between objects as though real photographs were provided as input.

Walking through a scene and taking pictures of it is certainly a task that almost everyone can carry out. In the photographer's metaphor, reference to the source domain isn't limited to taking snapshots. Introducing customization functions of the virtual environment—such as placing objects, setting a background, and so on—leads to a more realistic implementation of the metaphor. In fact, this also enables the more experienced photographer to change the scene's setup.

To support content editing, the virtual environment can be dynamically changed and customized by populating the scene with new, user-created objects and by interactively editing colors and textures. Users can choose colors and textures either from a predefined set of prototype patterns or extracted from retrieved images. In the case of textures, the 2D data is backprojected to the 3D world, which is dynamically modified according to the new features (see Figure 3).

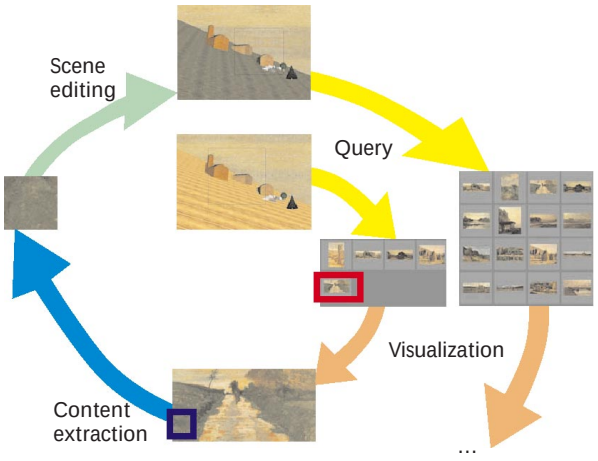


Figure 3. Color patterns can be extracted from retrieved images and added to the system's texture database. These patterns become available for customization through the texture palette (see Figure 1). Thus, it's possible to use the content of retrieved images to edit the scene in order to refine a query.

Images courtesy of De Luca Editore

Implementing the metaphor

Figure 1 shows an overview of the system's main functions. Those that apply to the 3D scene appear in the top part of Figure 1. The bottom part shows functions that apply to 2D images. These can be used to browse retrieved images and extract their content.

The system supports scene editing and navigation in the 3D world. With the query function the user takes photographs of the 3D world and uses them to retrieve similar images from a database. The system also enables visualization (in both low and high resolution) and content extraction from retrieved 2D images.

We embedded the 3D virtual environment into a 2D graphical user interface (GUI). A set of buttons lets users switch between the different operation modes. The buttons are labeled with icons representing the functionality they give access to.

Editing mode. Users customize and populate the 3D environment to reproduce a scene with a high degree of realism. The 3D environment includes predefined as well as user-defined 3D models, which users can specify through 3D editing tools. In this way, the environment is modified to replicate, at best, features of

realistic scenes. Objects can be taken from a palette, then placed in the scene by pointing at the object's desired final position. Users may also select an object in the scene and drag it around to place it somewhere else or remove it completely from the scene.

In addition, users interactively modify the 3D scene by changing object color and texture. On the rightmost panel of the system interface buttons permit the selection of the distinct operation modes. Mode-dependent tools appear on the upper panel. The buttons let users select one of the three submodes (object, color, and texture editing) and show their respective palettes (see Figure 4). Users may choose a color or a texture from a palette and apply it to any object in the scene. Textures can also be extracted from retrieved images and applied to objects in the scene. Textures render more realistic computer generated scenes[12,13] to enhance the perception of movement in the 3D scene and to provide a sense of depth.

Navigation and querying modes. Navigation in the virtual world may take place either continuously or by jumping from one location to another, using bookmarks. In the continuous mode, navigation occurs through a walk metaphor that lets users move freely. Additional interaction schemes such as panning and tilting let users fine-tune the viewpoint's position and orientation.

Special buttons permit the use and definition of bookmarks. A well-known metaphor, bookmarks are commonly available in browsers to navigate the World Wide Web. In our system, bookmarks note interesting viewpoints in the scene and can be used to jump from one location to another. This approach saves time in moving between distant points in the scene.

In querying mode, users extract a 2D example from the 3D scene according to the photographer metaphor. Users observe the scene, decide the area to capture as an example for querying, set the viewpoint and orientation, and take a picture. In this way, the 3D scene is automatically projected onto a 2D image mirroring the spatial arrangement of scene objects according to the user's point of view. Just as in the real world where the camera takes a portion of the landscape and projects it onto the film, in the virtual world the graphics engine projects a part of the 3D scene in the frame buffer to generate a bitmap image. Actually, the photographer metaphor is easier to deal with than real-world photography because users don't have to worry about technical aspects such as lighting or field depth. The resulting picture is fed to the retrieval engine, which extracts the interesting features out of it, runs the query, and shows the results in a separate 2D window.

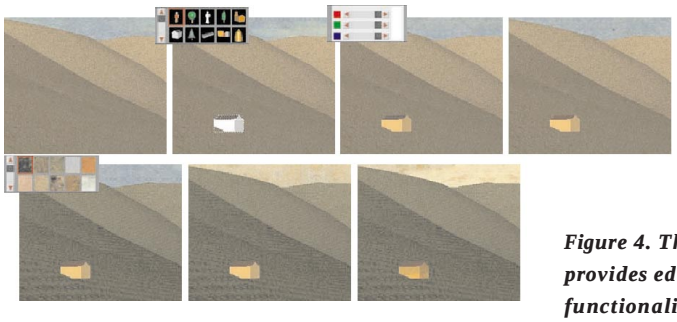


Figure 4. The system provides editing functionalities to customize the 3D scene. The tools available allow users to insert, remove, and displace objects as well as change their color and/or texture.

The 2D results window shows low-resolution thumbnails of the retrieved images. The images are listed with similarity ranking scores decreasing from left to right and from top to bottom. Full-resolution images are also available for detailed user inspection. Users can extract color textures from the displayed images. Textures can be added to the texture database and may later be used in editing mode to change an object's appearance.

Image retrieval based on color and structure

The query photograph feeds a retrieval engine, which selects images with content similar to the query. This requires extracting content descriptors from database images during archiving and using them to compute a content similarity score during retrieval.

Visual features extracted from images at archival time include color and edge orientation. To preserve the spatial localization of these visual features, each image in the database is partitioned into a predefined number n of rectangular tiles (presently $n = 9$). For more detailed insight on the techniques we used to extract color and shape features, see the sidebars "Extracting Color Features" and "Extracting Shape Features" (next page) respectively.

Histograms represent the visual content of each tile both in terms of color and edge orientation. Representing both classes of features through histograms enables a uniformity of color and edge data structures within the system. Histograms also allow for a multiresolution description of images. Given a partitioning in n rectangular tiles, a generic region S can be approximated as the union of $m \leq n$ tiles.

For the purpose of indexing and retrieval, a symmetric *distance matrix* $A \in \mathrm{R}^{n \times n}$ is introduced.

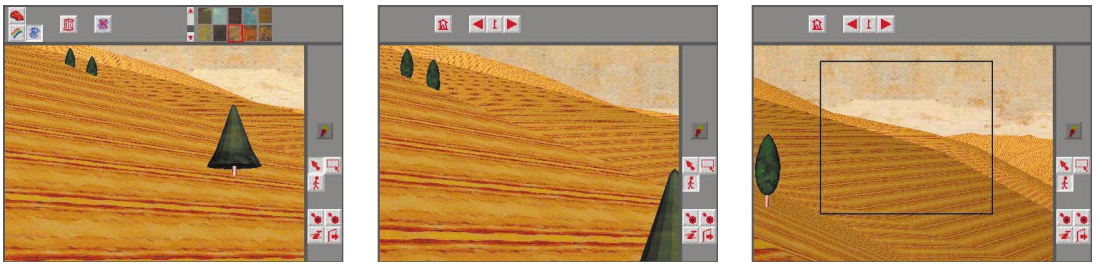
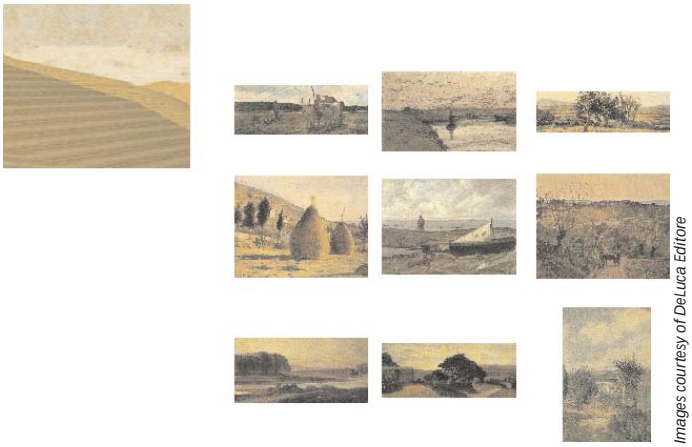


Figure 5. The user navigates the virtual environment until he discovers an interesting location. The last image of the sequence shows the photograph taken by the user to query the image database (see Figure 6).



Images courtesy of DeLuca Editore

Figure 6. On the left, the photograph used to query the engine. It features yellowish colors, somewhat darker in the foreground. The panel on the right shows the result set. The system determines the ranking according to the similarity of the tiles that both query and database images are partitioned into.

Extracting Color Features

Given a generic image I, and its partitioning in $n = x \times y$ ($x = 3$, $y = 3$), the image's chromatic content is expressed through a set of n histograms $\{Hi(I)\}_{i=1}^{n}$, the i-th histogram $H_i(I)$ representing the chromatic content of the i-th tile of image I.

To compute the color histograms, the color space is partitioned into 39 regions. Given the HLS (hue, lightness, saturation) representation of the color space, we selected nine hues with two different lightness and saturation values. In addition, three different grays (100, 50, and 0 percent) were added to the reference set. Given an image I, its histogram is computed by mapping each original color into one of the 39 reference colors. Mapping is performed by computing color distances in the $L^*u^*v^*$ space.[13] This choice has positively influenced the effectiveness of color-based image retrieval because in that space the Euclidean distances are guaranteed to conform to perceptive color distances.[14]

At archival time the histograms are computed for all the images added to the repository.

The distance between two histograms H and H' is computed as follows:

$$D(H, H') = (H - H')^T A(H - H') = \sum_{i=1}^{n}\sum_{j=1}^{n}(h_i - h_i')(h_j - h_j')a_{ij}$$

Each element a_{ij} of the distance matrix weights the extent to which the difference between the i-th and j-th bins contributes to D. If A is a diagonal matrix, only differences between corresponding bins contribute to the computation of D. However, if A isn't a diagonal matrix, elements a_{ij} ($i \neq j$) can be used to model the cross-distance between noncorresponding elements. This is particularly useful to partly recover from the loss of information associated with the discrete nature of content representation through histograms.

Two distinct distance matrices (A_C and A_S) are defined for the computation of the distance between color histograms and edge orientation histograms, respectively. The color distance matrix A_C is a 39×39 symmetric matrix whose elements a_{ij} encode the perceptual similarity between the i-th and j-th reference color. The structure distance matrix A_S is an 8×8 symmetric matrix whose elements a_{ij} encode the perceptual similarity between the i-th and j-th reference orientation.

When submitting a query image to the retrieval engine as an example, the system extracts visual features (color and structure) as explained above. The system uses the resulting histograms for the query image to look up the best matching images in the repository.

Querying by photographs

We implemented the system for querying by photographs in C++ with the use of the OpenGL library. It currently runs on an SGI Octane (MIPS R10000) with the Irix 6.4 operating system.

At archival time, the system automatically extracts color and edge orientation features from images to be added to the database. It also inserts the color and edge orientation histograms into an

Extracting Shape Features

To extract structure information, each image I in the database is first normalized so that the maximum dimension is set to 128 pixels and the other one is set to keep the aspect ratio of the original image. Then, gradients for RGB intensities in x and y directions ($\delta^x_{i,j}$, $\delta^y_{i,j}$) are computed for each pixel $p_{i,j}$ in the normalized image:[2]

$$\delta^x_{i,j} = \frac{p_{i+1,j} - p_{i-1,j}}{\sqrt{2(p^2_{i+1,j} + p^2_{i-1,j}}}$$

$$\delta^y_{i,j} = \frac{p_{i,j+1} - p_{i,j-1}}{\sqrt{2(p^2_{i,j+1} + p^2_{i,j-1}}}$$

Edge orientation and intensity are

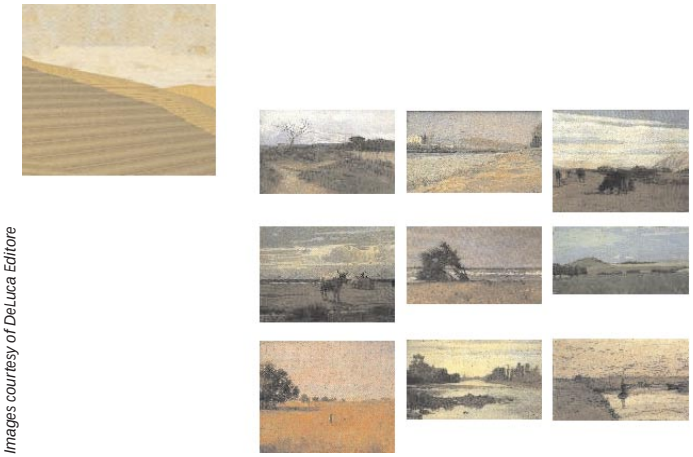
$$\vartheta_{i,j} = \tan^{-1}\left(\frac{\delta^x_{i,j}}{\delta^y_{i,j}}\right) \qquad p_{i,j}\sqrt{(\delta^x_{i,j})^2 + (\delta^y_{i,j})^2}$$

The result of the edge detection phase is a set of edges $\varepsilon = \{E_1, \ldots E_e\}$. Each edge E_i is described by a couple $\langle \rho, \vartheta \rangle$ where ρ is the intensity of the edge and ϑ its orientation. The value of the intensity is discretized into l ($l = 128$) levels and a histogram of intensities $M \equiv \langle m_1, \ldots m_l \rangle$ is computed. To describe a generic image's structural content, only relevant image edges should be used. To this end, a threshold τ discards all those image edges whose intensity doesn't exceed τ. The value of τ is computed such that the ratio between the number of preserved edges and the number of edges is about 15 percent:

$$\tau : \frac{\sum_{k=\tau}^{l} m_k}{\sum_{k=1}^{l} m_k} \simeq 0.15$$

Using the threshold τ, a new image is produced, preserving only orientation information about edges $E_i \equiv \langle \rho_i, \vartheta_i \rangle$ for which $\lfloor \rho_i \rceil \geq \tau$.

The resulting image is then partitioned into $n = x \times y$ rectangular tiles R_i ($i = 1, \ldots, n$). The orientation space is discretized into t ($t = 8$) levels and an orientation histogram $T(R_i)$ is computed for each tile R_i. The set of histograms $T = \{T(R_1), \ldots, T(R_n)\}$ describes the structure of image I.



Images courtesy of DeLuca Editore

Figure 7. The same photograph as in Figure 6 is also used to run a query based on structure. The main feature of the query image is the almost horizontal skyline. This feature is shared by the retrieved images in the result set shown on the right.

indexing structure. To provide textures and 3D models for customizing the virtual environment, two repositories are set up with textures and object models, respectively. Finally, an initial scenario, including some predefined bookmarks, is also provided.

Additional models, textures, and images may be added later. For instance, the texture database will progressively include textures that users extract from the retrieved images.

A sample session

After a short navigation (see Figure 5), the user takes a photograph of the landscape (picture on the left-hand side of Figure 6). The result images appear on the right-hand side of Figure 6.

The query image is characterized by yellowish colors, both for the ground and the sky. However, because of shading, the color at the bottom of the image is quite dark. Retrieved images feature either dark colors at the bottom or yellow color patterns in the central and/or top parts, thus matching, at least partially, the query image.

The user then runs the same query for images with a similar structure. Figure 7 shows the query and result images. Similarly to the query, retrieved images feature an almost horizontal skyline. For images with lower scores, some objects such as

Images courtesy of DeLuca Editore

Figure 8. The photograph on the left is used in a query based on color features. Images retrieved show a match with the query image either for the ground or for the sky, if not for both—as is the case for the first two images. Also note that the first ranked image features a ratio between regions occupied by the sky and the ground similar to that of the query image.

Images courtesy of DeLuca Editore

Figure 9. The query image features two objects—a tree and a wall—in the foreground. The first retrieved image is characterized by a similar structure, since it features both a vertical and horizontal object—the lady and the wall, respectively.

Figure 10. The structure of the query and the three best-ranked images of Figure 9 is shown through edge images. All of the retrieved images distinctly show a set of horizontal edges that account for the ranking of those images.

trees, animals, or hills, may stand out against the sky thus changing the silhouette.

Next, the user runs a new query by color. The user can change the environment through the system's editing functions. In Figure 8 the most remarkable changes the user made include a green color pattern for the ground and a color pattern featuring blue and white for the sky.

The retrieved images match the query. In particular, the first-ranked image shows a good match for all regions but the central, and central-right one (as previously stated, each image is partitioned into nine tiles).

Slightly changing the viewpoint results in a new scene, with a tree and a wall. Once more, the user changes color patterns in the environment and takes a new photograph to run a query by structure. Figure 9 shows the system's results and the user's query.

The first retrieved image matches the query image almost exactly. Because the system doesn't support high-level semantics, it can't distinguish between a tree and a human. However, a detailed inspection of the other images unveils several commonalities with the query image. In fact, as the edge images in Figure 10 show, the second and third images feature horizontal bands in the middle part.

Conclusion

In this article we presented a new interaction paradigm for querying image databases. The paradigm relies on a metaphor for taking photographs that most users are familiar with. As with real photography, the metaphor is scalable, ranging from a simple walk-and-click to a more professional, studio-like approach, which permits precise setups of the scene to be captured. Finally, the system presented here implements a relevance

feedback mechanism for query refinement.

To prove the effectiveness of the photographer's metaphor, we tested it on a content-based image retrieval engine that supports querying either by structure or by color.

The results reported in this article show the feasibility of the proposed approach and should stimulate us, as well as the whole research community, to investigate novel techniques for query specification in the domain of content-based retrieval of visual media. MM

Acknowledgments

We'd like to thank De Luca Editore of Rome for allowing us to use images from the following books: E. Spalletti, *Gli Anni del Caffè Michelangelo* [*The Years of the Caffè Michelangelo*], 1985; R. Monti, *Signorini e il Naturalismo Europeo* [*Signorini and European Naturalism*], 1984; R. Monti, *Le Mutazioni della "Macchia"* [*Mutations of the "Macchia"*], 1985; and D. Durbè, *Fattori e la Scuola di Castiglioncello* [*Fattori and the School of Castiglioncello*], 1983.

References

1. A. Del Bimbo, *Visual Information Retrieval*, Morgan Kaufmann, San Francisco, 1999.
2. K. Hirata and T. Kato, "Query by Visual Example: Content-Based Image Retrieval," *Proc. Advances in Database Technology—EDBT 92*, Lecture Notes on Computer Science, A. Pirotte, C. Delobel, and G. Gottlob, eds., Springer-Verlag, Berlin, Vol. 580, 1992, pp. 56-71.
3. A. Del Bimbo, M. Campanai, and P. Nesi, "Three-Dimensional Iconic Environment for Image Databases Querying," *IEEE Trans. Software Engineering*, Vol. 19, No. 10, Oct. 1993, pp. 997-1011.
4. L. Cooper and R. Shepard, "Turning Something Over in the Mind," *Scientific American*, Vol. 251, No. 6, Dec. 1984, pp. 114-120.
5. J. Hildebrandt and K. Tang, "A Two- and Three-Dimensional Ship Database Application," *Intelligent Image Database Systems*, S.K. Chang, E. Jungert, G. Tortora, eds., World Scientific, Singapore, 1996.
6. V.N. Gudivada, "Spatial Knowledge Representation and Retreival in 3D Image Databases," *Int'l Conf. Multimedia and Computing Systems*, IEEE CS Press, Los Alamitos, Calif., May, 1995, pp. 90-98.
7. E. Ciabatti et al., "Towards a Distributed 3D Virtual Museum," *Proc. Int'l Working Conf. on Advanced Visual Interfaces (AVI 98)*, ACM Press, New York, 1998, pp. 264-266.
8. A. Müller et al., "Towards the Virtual Internet Gallery," *Proc. 6th Int'l Conf. Multimedia Computing and Systems (ICMCS 99)*, IEEE CS Press, Los Alamitos, Calif., 1999, pp. II.216-219.
9. M. Leissler, M. Hemmje, and E. Neuhold, "Supporting Image Retrieval by Database Driven Interactive 3D Information Visualization," *Proc. 3rd Int'l Conf. Visual Information Systems (Visual 99)*, Lecture Notes in Computer Science, N. Huijsmans and A. Smeulders, eds., Springer-Verlag, Berlin, 1999, pp. 1-14.
10. G. Lakoff and M. Johnson, *Metaphors We Live By*, University of Chicago Press, Chicago, 1980.
11. H. Martin, *A Computational Model of Metaphor Interpretation*, Academic Press, San Diego, 1990.
12. P. Haeberli and M. Segal, "Texture Mapping as a Fundamental Drawing Primitive," *Proc. 4th Eurographics Workshop on Rendering*, M. Cohen, C. Puech, and F. Sillion, eds., 1993.
13. J.D. Foley et al., *Computer Graphics: Principles and Practice (Systems Programming)*, 2nd ed., Addison-Wesley, Reading, Mass., 1996.
14. J. M. Corridoni, A. Del Bimbo, and P. Pala, "Retrieval by Color Semantics," *ACM Multimedia Systems*, Vol. 7, No. 3, 1999, pp. 175-183.

Jürgen Assfalg is a PhD candidate carrying out his research activity at the Department of Systems and Informatics at the University of Florence, Italy. He received an MS in electronics engineering from the University of Florence in 1998. His main research interest is advanced visual interaction, including multimedia, 3D graphics, and image databases.

Pietro Pala is an assistant professor in the Department of Systems and Informatics at the University of Florence, Italy. He received his MS in electronic engineering at the University of Florence in 1994. He received a PhD in information science from the same university in 1998. His current research interests include pattern recognition, image and video retrieval by content, and related applications.

Readers may contact the authors at the Department of Systems and Informatics, University of Florence, Via S.Marta 3, 50139, Florence, Italy, e-mail {assfalg,pala}@dsi.unifi.it.