This is a preprint version of the following article, published by Springer (doi:http://dx.doi.org/10.1007/s11042-013-1550-5): R. Yus, S. Ilarri, and E. Mena, "*Real-time Selection of Video Streams for Live TV Broadcasting Based on Query-by-Example Using a 3D Model*", Multimedia Tools and Applications, ISSN 1380-7501 (print version), ISSN 1573-7721 (electronic version), Springer, 2013.

Real-time Selection of Video Streams for Live TV Broadcasting Based on Query-by-Example Using a 3D Model

Roberto Yus[a], Sergio Ilarri[a], Eduardo Mena[a]

[a]*University of Zaragoza, María de Luna 1, Zaragoza, Spain*

Abstract

The emergence of low-cost cameras with nearly professional features in the consumer market represents a new important source of video information. For example, using an increasing number of these cameras in live TV broadcastings enables obtaining varied contents without affecting the production costs. However, searching for interesting shots (e.g., a certain view of a specific car in a race) among many video sources in real-time can be difficult for a Technical Director (TD). So, TDs require a mechanism to easily and precisely represent the kind of shot they want to obtain abstracting them from the need to be aware of all the views provided by the cameras.

In this paper we present our proposal to help a TD to visually define, using an interface for the definition of 3D scenes, an interesting sample view of one or more objects in the scenario. We recreate the views of the cameras in a 3D engine and apply 3D geometric computations on their virtual view, instead of analyzing the real images they provide, to enable an efficient and precise real-time selection. Specifically, our system computes a similarity measure to rank the candidate cameras. Moreover, we present a prototype of the system and an experimental evaluation that shows the interest of our proposal.

Keywords:
Query by example, Camera shot similarity computation, User interfaces to manage 3D scenes

Email addresses: ryus@unizar.es (Roberto Yus), silarri@unizar.es (Sergio Ilarri), emena@unizar.es (Eduardo Mena)

1. Introduction

Nowadays the consumer market offers cameras with very interesting features at low prices. These cameras offer a good image quality, they can be remotely controlled, etc. So, they can be used in TV broadcastings combined with professional cameras. This is very interesting for broadcasting corporations because they are focusing their efforts on the enrichment of their offers but at the same time trying to reduce their production costs. Thanks to the low cost of these cameras, many of them can be used, for example in the live broadcasting of a sport event. This allows an enrichment of the content consumer experience, as they can provide new kinds of shots, without having a big impact on the total number of cameramen required. However, the higher the number of cameras employed in a live broadcasting, the more difficult the selection of the camera that must be broadcasted.

Among the many tasks of a Technical Director (TD), the person responsible for the content production, the selection of the cameras is probably the most challenging one. He/she has to be aware of all the shots that the cameras in the scenario are providing to select the best one and broadcast it. Depending on the situation the TD could be interested in certain specific shots of the objects involved. For example, the TD in charge of the live broadcasting of a car race could be interested, at the last lap of the race, in a shot that shows the cars fighting for the first position. A system to provide automatically the TD with the cameras that are obtaining these interesting shots (as in [1, 2, 3, 4]), would be very useful for broadcasting corporations. However, a challenge for this kind of systems is how to obtain a precise definition of what an interesting shot for the TD is, to query the available cameras in order to retrieve those that can provide a similar shot. Both [1] and [3] are focused on the automatic selection of the best camera view in ball sports without interaction of the TD, and specifically they considered that an interesting shot is the one that obtains the best view of the ball and the one that provides a clear view, respectively. In [2] the use of location-dependent queries [5] was proposed as the basic building block for monitoring sport events and help the TD in the broadcasting task; however, the system had several limitations, as it was not able to manage 3D models of the queries and the scenes, and occlusions among objects were not taken into account. In [4] the TD can input his/her desired shot in terms of the specific objects involved (by selecting their name from a list), the minimum percentage of them that must be covered by the camera (as a percentage), the kind of view that the camera must obtain of the object (selecting between front/rear, top/bottom, and/or right/left side), etc. However, defining

these parameters is not always easy, and so it would be interesting to have instead a more intuitive mechanism, such as a visual interface to define a sample shot.

The problem of retrieving information using an example defined by the user is known in the literature as *Query-By-Example* (QBE) [6] and was initially conceived for querying relational databases about numeric values or text. However, due to the rapid growth in the number and size of image databases, *Query-By-Image* (QBI) [7] was introduced[1]. To query-by-image a multimedia source, the user selects an image that is used as a prototype/template of the target. Then, low-level features (such as information about the color distribution, texture, shapes, etc.) of the example image and other images are obtained to compute their similarity. The problem of QBI is to find the proper query image that precisely represents the scene that the user has in his/her mind. To overcome this, several alternatives have been proposed. So, *Query-By-Sketch* [8] allows the user to sketch/paint the example image. However, it is difficult for a user to draw a precise example shot by hand, and therefore the accuracy of the retrieval is low. To avoid this problem, in works like [9] the user defines the example shot using "3D interfaces exploiting navigation and editing of 3D virtual environments", which allows a very precise definition of the type of image required. With both the approaches based on sketches or the ones using a 3D interface, once the example shot is defined a traditional QBI approach is performed.

In this paper we present our proposal to help a TD to visually define an interesting example shot by using an interface for the definition of 3D scenes. The TD easily express the shot he/she wants to obtain and this information is used to continuously query-by-example the available cameras to obtain those that can provide a similar shot[2]. One important difference with existing works is that, instead of applying traditional image processing techniques over 2D projections of 3D query scenes and the candidate images, we propose obtaining high-level features (related to the semantics of the specific objects in the scene) directly from the 3D scenario. In this way, information such as the specific objects in the scene, the percentage of each object shown, the percentage of the scene filled by each object, the specific viewpoint of each object shown, etc., can be obtained precisely and in *real-time* (in our prototype the 3D model of the scenario is refreshed

[1] QBI is QBE applied to Content-Based Image Retrieval (CBIR), by using an image as a query.

[2] Broadcasters are now beginning to use 3D stereo cameras, which can be represented as two traditional cameras (one per lens) in our approach, as considering the special capabilities of 3D stereo images is out of the scope of this paper.

once per second, so the computation time should not exceed one second); for example, processing techniques over real images may find it very difficult to detect the identity of certain objects (e.g., the "Kaiku" team rowing boat vs. any other boat) or even the types of objects due to the well-known problem of the "semantic gap" between the visual features and the richness of human semantics [10]. We present a similarity function that uses the high-level information to measure the similarity of the views provided by the available camera sources compared to the user query image. In addition, we have developed a prototype to evaluate our proposal and carried out tests with users. The results obtained show the interest of our approach.

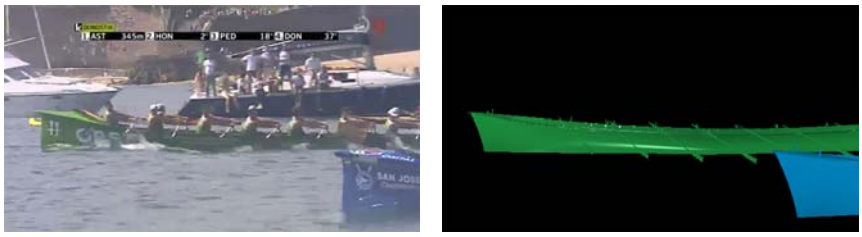
The rest of this paper is structured as follows. In Section 2 we explain the use case we have considered and the context of our proposal. In Section 3 we present the basics of our approach by introducing the sample shot definition and its processing to extract high-level features. In Section 4 we explain the method that we have developed to measure the similarity between two images. In Section 5 we show the tests performed to evaluate our proposal. In Section 6 we review some related works. Finally, conclusions and future work are included in Section 7.

2. Context

As an example of a live TV broadcasting we consider a specific sport event: the rowing races of San Sebastian (Spain), which are very popular and attract a lot of attention in the north of Spain. In this scenario the broadcasting corporation combines professional cameras with low-cost remotely controlled cameras. For example, each rowing boat is equipped with a low-cost camera and professional cameras are placed in key positions in the scenario (on board other boats, on a helicopter, along the promenade, etc.). Besides, each rowing boat is equipped with a GPS transmitter and a compass that help the judges to determine the distance between the boats and the location of a boat with respect to its lane.

As in all live broadcastings, the TD has to make quick decisions to select the camera video stream to broadcast among the available sources. Depending on his/her decisions, the content broadcasted will be more attractive from the audience perspective. In [4], we presented a system to help in the challenging task of selecting a camera in live. The system is based on the use of an updated 3D model of the scenario. We proposed generating and updating this 3D model using the information available about the objects and cameras involved (mainly location information provided by GPS receivers on the rowing boats). Thanks to the 3D model, the system is able to obtain in real-time high-level features of each specific

object in the camera view by recreating them in a 3D engine (see Figure 1, where only the interesting objects are recreated –in this case the rowing boats, which are represented with a different color depending on its team–) and applying several 3D computations (summarized in Section 3.3).



(a) (b)

Figure 1: Real camera footage (a) and the camera view recreated by our system in a 3D engine (b).

The interface of the system presented in [4] allows the TD to define the constraints that an interesting shot has to fulfill. So, in the example of Figure 2 the TD has shown interest in obtaining "Any" camera that could provide a shot covering a certain rowing boat ("Kaiku"). At least "50%" of the boat has to appear in the shot, and the camera has to cover at least "75%" of the front view of the boat (a shot covering 100% of the front view is obtained by a camera located in front of the object pointing to it). This query can be defined quite quickly by the TD but is not very intuitive because such an interface does not facilitate the TD the task to clearly define the type of image of interest. Moreover, using this kind of definition, composing specific shots with several objects involved is very difficult. To overcome these problems, we propose in this paper an interface for the definition of 3D scenes that supports a precise definition of the query shot required. This interface, along with a novel and appropriate similarity function, will help the TD to easily define the shot that he/she wants to obtain. So, instead of defining the constraints that the shot has to fulfill, the TD shows exactly the kind of shot he/she wants and our system extracts the required constraints.

It is interesting to notice that the shots obtained at the same time instant by different cameras can be very similar regarding features such as the color distribution and that when the boats are shown from a large distance or partially visible it is hard to identify their specific teams. Moreover, the processing has to be performed in real-time, and therefore the information of the available camera shots has to be processed as quickly as possible. Due to all these reasons, applying traditional image processing techniques that obtain low-level features of the camera

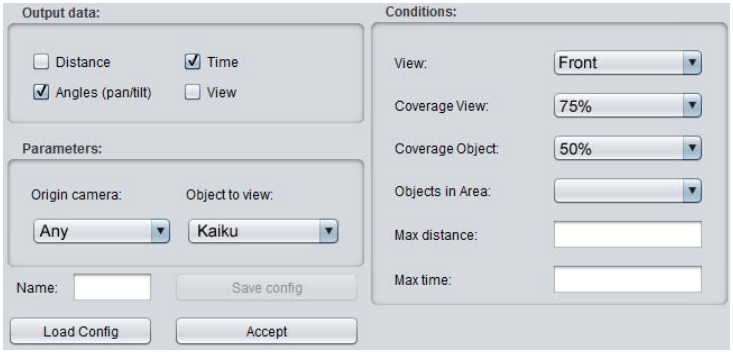


Figure 2: Example of low-level input form.

shots would not be enough to provide an accurate answer to the TD. So, we need a precise mechanism to compute the similarity between two shots in real-time based on other high-level information.

3. Obtaining Shots Similar to a 3D Sample Scene

The goal of our proposal is to help a TD to define interesting shots precisely and easily, use those example shots to query the available sources with a high refreshment frequency (in our prototype once every second), and compute their similarity to the images provided by the available cameras (see Figure 3). The architecture of our approach is based on three main tasks:

- "Definition of the query by the user": a user that interacts with the system defines an example shot using the "Example Shot Definition Interface" (explained in Section 3.1). The example shot will be processed to obtain its features (the specific objects inside, the percentage of them visible, their viewpoint, etc.) by the "Scene Analysis" module (explained in Section 3.3).

- "Real-time processing of the camera views": by using a 3D model of the scenario, our approach processes in real-time the views that the cameras provide. First, the "3D Scene Generation" module uses the 3D model of the scenario and information about the cameras to recreate their view in a 3D engine (explained in Section 3.2). Then, the virtual camera view will be processed by the "Scene Analysis" module, as in the previous task. Finally, the "Similarity Computation" module (explained in Section 4) evaluates the similarity between the features extracted for each user query and each of the

camera views, and the measurements obtained are used to present a ranking of interesting shots to the TD.

- "Update of the 3D model": to obtain accurate results, the 3D model of the scenario has to be updated with information as recent as possible. For this task, the "3D Model Management" module periodically receives information about the objects in the scenario and updates the 3D model (explained in Section 3.2).

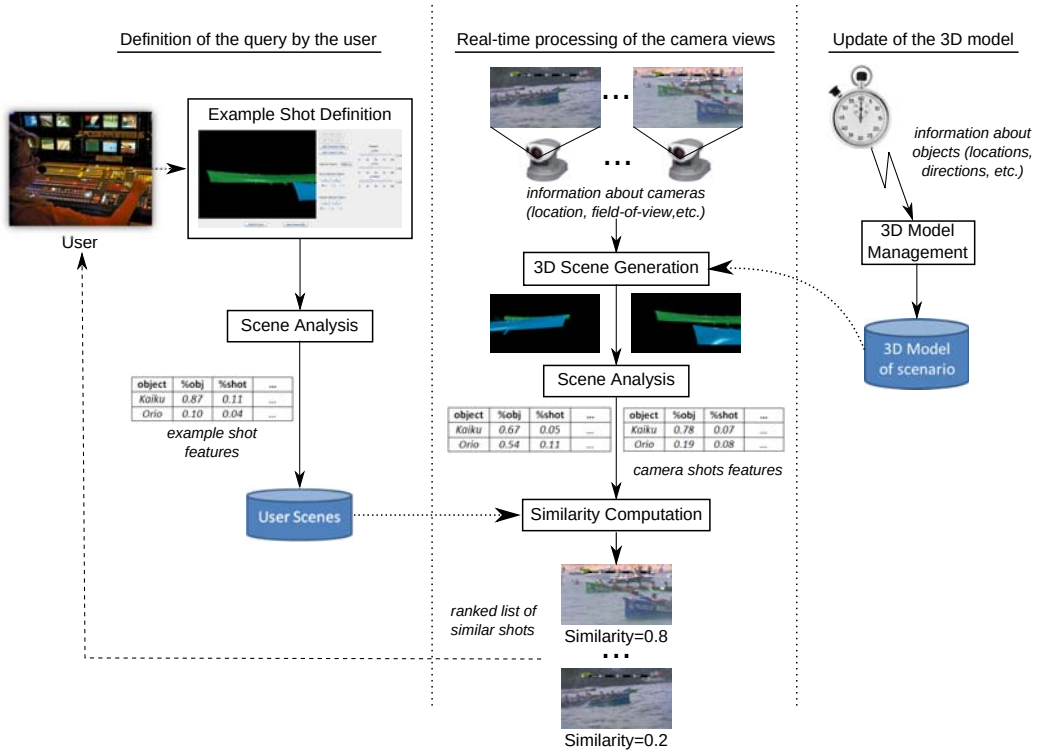


Figure 3: High-level architecture of the system and its three main tasks: definition of the query, processing of the camera views, and update of the 3D model.

As an example, in Figure 3 the TD defines a shot containing two specific rowing boats ("Kaiku" –the green boat– and "Orio" –the blue one–). Then, our approach processes the views provided by the cameras and determines that one of them provides a shot very similar to the TD's request, according to the features computed (see some of the features of the query –"example shot features"– and the camera views –"camera shot features"– in Figure 3).

3.1. Example Shot Definition

We propose the use of an interface for the definition of the example shot as a 3D scene. By using this kind of interface, the user is able to effectively represent an interesting scene [9] involving a specific view of one or more objects of the scenario. However, the selected interface has to fulfill some requirements. First, it must provide the user with a mechanism to add his/her own objects to the scene as well as mechanisms to edit it (i.e., move and rotate the objects), and to set the position and orientation of the virtual camera. Then, once the user has defined the 3D sample scene, the interface provides our system with the information shown in Figure 4. Where the identification of a certain object is a unique name, its orientation is given as the rotation angles for the three axis of the plane (i.e., heading, tilt, roll), and its distance to the camera is the distance between the centroid of the object and the camera (vector with three components, for the X, Y, and Z dimensions). In addition, the orientation of the camera is defined by its current horizontal and vertical rotation (i.e., pan and tilt) and its Field-of-View (FOV) is defined as the horizontal and vertical Angle-of-View.

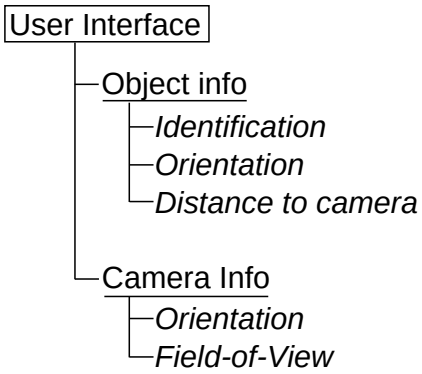


Figure 4: Information provided by the example shot definition interface to the system.

Any external 3D computer graphics software able to provide this information could be used as the interface of our system for modeling (e.g., *Autodesk 3Ds Max*, *Blender*, *SketchUp*, etc.), so we can exploit the functionality of external 3D software to define the scene in a comfortable way. As we explain later, in our prototype we have used a Java 3D engine (*JMonkey Engine*), both for the definition of the scenes and for the analysis of the camera views.

The use of interfaces for the definition of 3D scenes to query-by-example was analyzed in [9], where the tests with users showed that the time needed to define a complex query scene (involving the definition of a specific terrain and different

objects with textures, which are tasks not required in our approach) ranged between 10 and 100 seconds with the prototype presented in that paper. To facilitate the work of a TD, our approach enables him/her to pre-define and store queries that can be used as pattern queries. So, those queries can later be modified in real-time according to the current needs (e.g., by rotating/moving the virtual camera, moving/rotating the objects, changing the identification of the objects appearing in the scene, etc.). For example, a TD in charge of the broadcasting of a rowing boat race could pre-define (before the event) a pattern query with a full side view of any rowing boat and then modify this scene during the broadcasting by rotating the camera and indicating a specific identity for the boat (e.g., the one of the local team). Of course, ad hoc queries completely defined during the broadcasting are also supported.

3.2. 3D Model Management and Scene Generation

In order to process the views provided by the cameras in the scenario, we propose using an up-to-date 3D model of the scenario [4]. Using this 3D model we perform the following process for each camera in the scenario to obtain high-level features of their current views. First, we recreate the camera view in a 3D engine by using information about the current location and direction of the different objects and cameras (this information could be provided by a GPS and a compass, as in the experiments presented in Section 5). Then, we use the same scene analysis method used for the sample shot (see Section 3.3). Due to the highly-dynamic nature of the scenario where both objects and cameras move, and so the views of the cameras change dynamically, we continuously perform the previous process to obtain up-to-date information.

Obtaining in real-time the cameras that are able to provide the TD with the required view is possible due to the up-to-date 3D model (see Figure 3 on the right). In parallel with the processing of the user queries, the system efficiently keeps the 3D model updated with the information of the objects and cameras in the scenario (obtained from different sensors). It should be noted that this is not an overload for the system, as it only involves obtaining the interesting information and storing it, and it can even be performed on another computer.

3.3. Scene Analysis

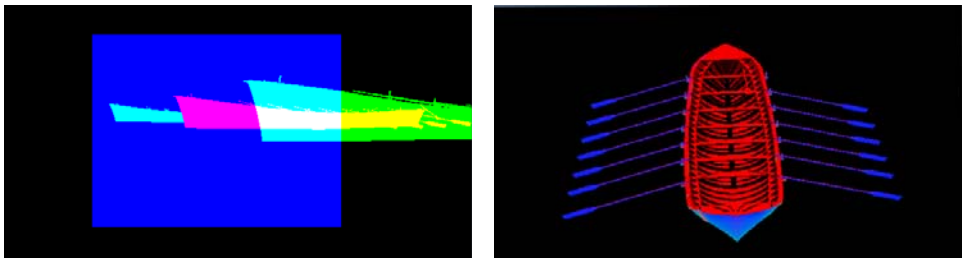
To obtain the information needed to compute the similarity between a shot defined by the user and each current camera shot of all the cameras in the scenario, we propose using the ideas we presented in [4]: recreate the scene in a 3D engine

and apply 3D computations to efficiently obtain (in real-time) the following high-level features for each object in the scene:

- Identification of the specific object.
- Percentage of the object visible (taking occlusions into account).
- Percentage of the image that the object fills.
- Location and orientation of the object in the image.
- For each viewpoint of the object (front/rear, top/bottom, right/left side):
  - Percentage of the viewpoint visible.
  - Percentage of the image that the viewpoint fills.

To obtain this information the system renders the 3D scene into 2D projections and analyzes them. However, as this is a time-consuming task we have to minimize the number of renderings needed. For this, the system uses different colors and transparencies (based on color blending) to obtain as much information as possible with each rendering. In this way, to obtain the percentage visible of a certain (target) object taking occlusions into account (caused by other objects or due to the fact that the object does not fit the camera FOV), the following steps are followed:

1. Apply a red color to that target object, a green transparent color to other objects in the scene, and a blue transparent color to the FOV of the camera (see Figure 5(a)).
2. Obtain a rendering of the 3D scene and analyze the color channel of every pixel in the following way:
   - A pixel that is red, blue, and green belongs to the object that the system is considering but is currently occluded by other objects.
   - A pixel that is red but not blue is a pixel of the object outside the FOV.
   - A pixel that is red and blue but not green is a pixel of the object visible in the camera shot.
3. Compute the percentage as the number of pixels of the object visible in the camera shot divided by the total number of pixels that belong to the object.



(a) (b)

Figure 5: Computation of the percentage of an object that a camera is viewing (a) and of the kind of view obtained of another object (b).

In the same way, to check the viewpoints of the object being viewed the system uses different colors and light sources, according to the following steps:

1. Set directional light sources to illuminate only the required viewpoints (for example, in Figure 5(b) a red and a blue light sources illuminate only the top and rear views of the object, respectively).
2. Obtain a rendering of the 3D scene and analyze the color channel of every pixel. In Figure 5(b) red pixels belongs to the top view and blue pixels to the rear view[3].
3. For each viewpoint move and rotate the camera to cover it completely. Then, obtain and analyze a rendering.
4. Compute the percentage by comparing the number of pixels of each color in the different renderings.

In this way, depending on the information that has to be obtained from a 3D scene, the number of renderings could change. In the best case (e.g., to obtain only the percentage visible of an object), there will be a single rendering per object. In the worst case (e.g., to obtain all the information possible), there will be up to ten renderings per object[4].

With the high-level features extracted from the scene defined by the user and the views of the cameras, the next step is to obtain a similarity measurement for

[3]Notice that the paddles are mainly shown in blue as they are considered as belonging to the rear view of the object.

[4]One rendering to obtain the percentage visible of the object and the percentage of the shot filled by it, and nine additional renderings to obtain the kind of view of the object (the system needs one rendering per view covering it completely –up to 6 in total–, and 3 renderings more to compute the percentage of each view covered).

each camera view. Thus, the different camera shots currently available can be automatically ranked in decreasing order of similarity between the view they can provide and the query image. In Section 4 we explain the method developed to obtain the similarity between the example shot defined by the user and a shot provided by a camera.

3.4. Query-by-Example 3D Prototype

We have developed a prototype that implements the ideas presented along this section[5] and uses the similarity computation approach defined in Section 4. It enables the user to define a sample shot and obtain the ranked list of the current camera views according to their similarity (see Figure 6). The main components of the prototype are:

- *Shot definition interface*: it allows the user to populate the 3D scene with different objects, move/rotate them, and navigate the scene.
- *Result set display*: it shows the camera views ranked according to their similarity to the user query.

Notice that the prototype enables the user to modify the different weights assigned to the features that our approach considers (see Section 4.4). In this way, the user can refine the ranking criteria by using the sliders shown on the right of the shot definition interface. The prototype has been developed as a *Java* program and makes use of a powerful and free Java 3D engine called *JMonkeyEngine*[6]. The 3D models of the objects represented (in our case study, rowing boats) have been obtained and modified using *SketchUp*[7]. Finally, a *MySQL* database has been used to store the updated 3D model of the scenario with the information of all the objects and cameras involved.

4. Measuring the Similarity

To measure the similarity between the query image I_q defined by the user and a candidate image I, we propose the following formula, that makes use of the high-level features extracted from both images to take into account the similarity of each specific object appearing in the two pictures:

[5]The prototype is available at `http://sid.cps.unizar.es/MultiCAMBA/QBE`.
[6]See `http://jmonkeyengine.com`.
[7]See `http://www.sketchup.com`.

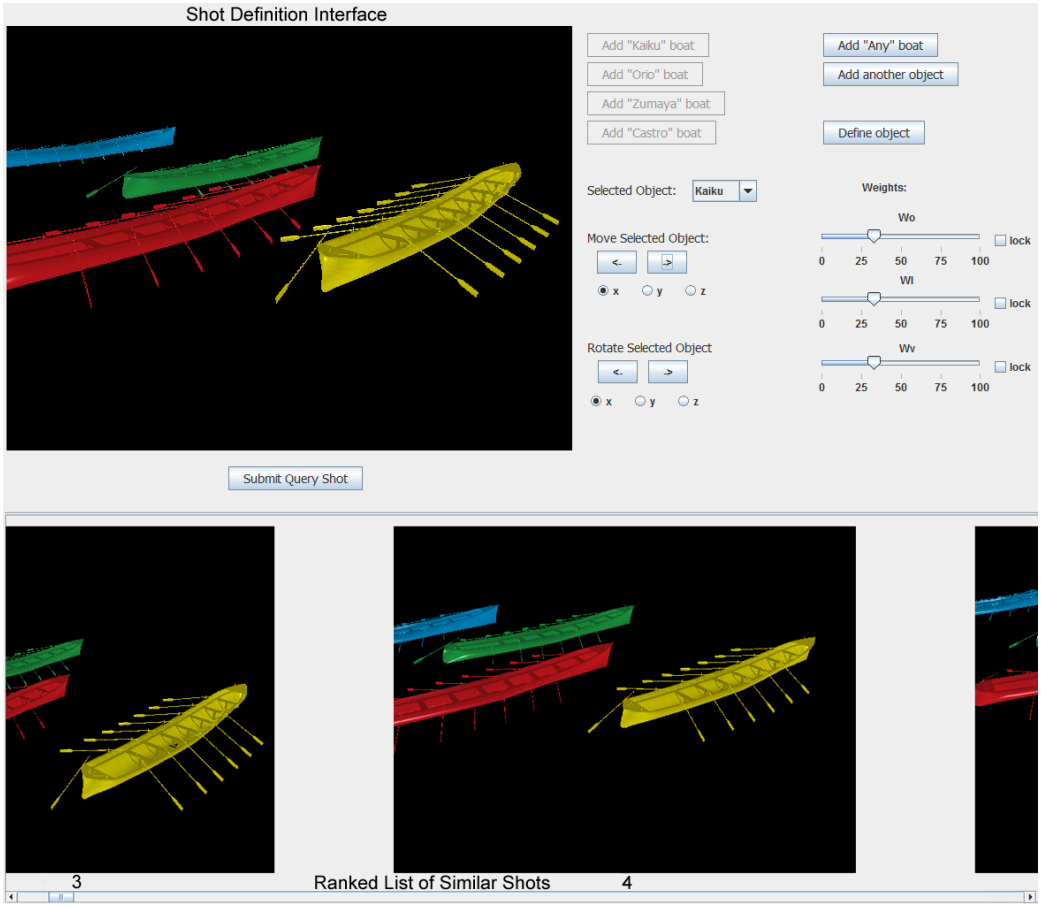


Figure 6: Snapshot of the Query-by-Example 3D prototype.

$$S(I_q, I) = \sum_{i=1}^{n} \gamma_{o_i} S_{object}(o_i, I_q, I) \tag{1}$$

$$\gamma_o = \frac{pct_{img}(o, I_q)}{\sum_{i=1}^{n} pct_{img}(o_i, I_q)} \tag{2}$$

where n is the number of objects, $S_{object}(o, I_q, I)$ is the similarity between the two images if only the object o is considered, and γ_o is the weight (relative importance) assigned to this object in the query image I_q defined by the user.

To avoid overwhelming the user by requesting him/her to enter the weight for each object, we advocate using an objective value extracted from the scene. So, we consider that the user defines indirectly the importance of each object in the scene by adjusting the percentage of the image occupied by each of them. So, as shown in Formula 2, γ_o represents the percentage of the image filled by object o in the query image I_q ($pct_{img}(o,I_q)$) relative to the part of the image filled with objects ($\sum_{i=1}^{n} pct_{img}(o_i,I_q)$); this percentage is computed regarding only the part of the image filled with objects (as we are interested here in identifying the relative importance of the objects, rather than the amount of space they occupy in the image). For example, if we consider the upper-left image of Figure 6 as the query image defined by the user, the yellow and red objects (close to the camera) are clearly expected to be more important for the user than the other objects because they fill a greater amount of the shot.

In the rest of this section, we explain the different factors that define the computation of $S_{object}(o,I_q,I)$ for a given object o, a query image I_q, and an image I.

4.1. Differences in the Percentage Visible of Each Object

The first aspect that we consider to obtain the similarity of an object in two images is the percentage of it that is being covered and the percentage of the image that it fills. In this way, we compare the percentage of the object o visible in the query image I_q defined by the user with the percentage visible in the candidate image I, as well as the percentage of each image (I_q and I) occupied by o:

$$S_{pctObj}(o,I_q,I) = 1 - (\omega_{oi}\frac{\Delta pct_{obj}(o,I_q,I)}{pct_{obj}(o,I_q)} + \omega_{oii}\frac{\Delta pct_{img}(o,I_q,I)}{pct_{img}(o,I_q)}) \quad (3)$$

$$\Delta pct_x(o,I_q,I) = \min(pct_x(o,I_q), |pct_x(o,I) - pct_x(o,I_q)|),\ with\ \ x = \{obj,img\} \quad (4)$$

where in relation to the first addend $pct_{obj}(o,I)$ and $pct_{obj}(o,I_q)$ obtain the percentage of the object visible, taking occlusions into account, in image I and I_q, respectively; in relation to the second addend, $pct_{img}(o,I)$ and $pct_{img}(o,I_q)$ obtain the percentage of the corresponding images filled by the object. In addition, we normalize the difference between these percentages ($\Delta pct_{img}(o,I_q,I)$ and $\Delta pct_{obj}(o,I_q,I)$) to obtain a value between 0 and 1 that measures their similarity. Notice that, as we want to obtain an objective measurement, we consider that,

for example, given the percentage visible of an object in the query image x%, an image that shows $(x+y)$% is as similar as an image that shows $(x-y)$%. In this way, an image that does not show the object (i.e., it shows $(x-x)$% of it) is considered as similar as one that shows $(x+x')$% with $x \leq x' \leq 1.0$ (percentages are expressed here as values between 0 and 1); this is the motivation for the use of the min operator in Formula 4. Moreover, we assign each factor a weight ω_{oi} and ω_{oii}, and to preserve the objectivity we consider that $\omega_{oi} = \omega_{oii} = 1/2$.

4.2. Differences in the Viewpoint of Each Object

The second aspect that we consider is the kind of view obtained of the object in the two images. As part of the high-level features of a scene we consider that the following views of an object can be obtained: top/bottom, front/rear, and left/right side. So, the following formula defines the similarity of an object in two images according to the kind of view obtained:

$$S_{views}(o, I_q, I) = \sum_{i=1}^{n} \gamma_{v_i} (1 - (\omega_{vi} \frac{\Delta pct_{view}(v_i, I_q, I)}{pct_{view}(v_i, I_q)} + \omega_{vii} \frac{\Delta pct_{img}(v_i, I_q, I)}{pct_{img}(v_i, I_q)})) \quad (5)$$

$$\gamma_v = \frac{pct_{img}(v, I_q)}{\sum_{i=1}^{n} pct_{img}(v_i, I_q)} \quad (6)$$

$$\Delta pct_x(v, I_q, I) = \min(pct_x(v, I_q), |pct_x(v, I) - pct_x(v, I_q)|), \; with \;\; x = \{view, img\} \quad (7)$$

where n is the number of views, v is the vector that contains the views to check, whose components belong to the set {front, rear, top, bottom, left, right}, and v_i represents the view in position i of v. For each view of an object (front, rear, top, bottom, right side, and left side) we compute the similarity according to the percentage of the view obtained ($pct_{view}(v_i, I)$) and the percentage of the image filled with this view ($pct_{img}(v_i, I)$). We normalize the difference between these percentages ($\Delta pct_{view}(v_i, I_q, I)$ and $\Delta pct_{img}(v_i, I_q, I)$) to obtain a value between 0 and 1. We consider that the percentage of the view obtained and the percentage of the image filled with this view have the same importance to compute the similarity of an object (so, in our prototype the weights used for them are $\omega_{wi} = \omega_{vii} = 1/2$). Moreover, we assign each view a weight γ_v according to their importance in the image. For example, in Figure 6, regarding the yellow boat, the user seems to be interested in a top and rear view of such a boat.

4.3. Differences in the Location of Each Object

The location of each object within the image is an important parameter to take into account when computing the similarity. We consider the location of the camera and the object in the 3D scene to measure this factor. In particular, we use the angle defined by the bisector of the FOV of the camera and the vector defined by the location of the camera and the centroid (of the volume) of the object. For the sake of clarity, we further decompose the vector in terms of its horizontal and vertical components. To illustrate this, Figure 7 shows the different angles involved in the horizontal plane of two scenes (it would be similar for the vertical plane).

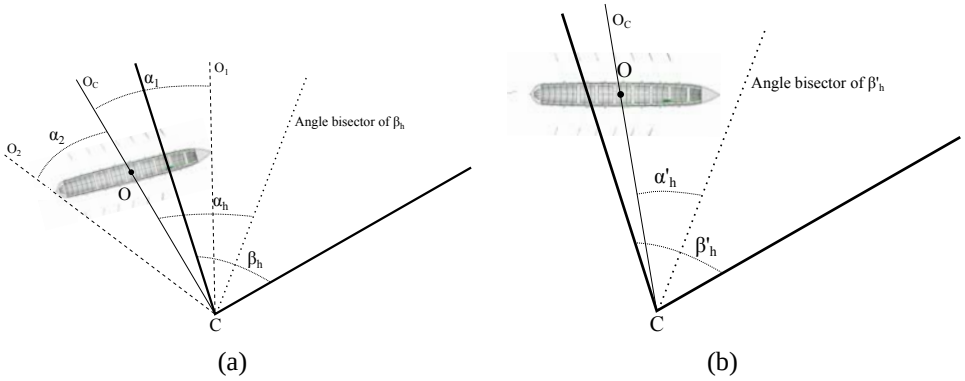


Figure 7: Horizontal angles involved in the computation of the similarity between two images (a) and (b).

So, we define the similarity of an object in two images according to its location as:

$$S_{location}(o, I_q, I) = 1 - (\omega_{li}\frac{\Delta\alpha_h}{max_h} + \omega_{lii}\frac{\Delta\alpha_v}{max_v}) \tag{8}$$

$$max_h = \max(\frac{\beta_h}{2} + |\alpha_1|, \frac{\beta_h}{2} + |\alpha_2|) - \varepsilon \tag{9}$$

$$max_v = \max(\frac{\beta_v}{2} + |\alpha_3|, \frac{\beta_v}{2} + |\alpha_4|) - \varepsilon \tag{10}$$

$$\Delta\alpha_x = \min(max_x, |\alpha'_x - \alpha_x|),\ with\ \ x = \{h, v\} \tag{11}$$

where α_h and α_v are the angles that define the location of the object in the horizontal and vertical plane of the sample scene I_q defined by the user, respectively.

Similarly, α'_h and α'_v measure the location for the scene we are comparing I_q with (i.e., image I). Besides, $\Delta\alpha_h$ and $\Delta\alpha_v$ stand for the difference in the location of the object in both scenes in the horizontal and vertical plane, respectively. As before, we consider the absolute value of these differences and we normalize them by using max_h and max_v (the maximum value of α_h and α_v for an object to be inside the FOV of a camera). In Formula 9 and 10, ε is a small value greater than 0, which must be subtracted from $\max(\frac{\beta_h}{2}+|\alpha_1|, \frac{\beta_h}{2}+|\alpha_2|)$ and $\max(\frac{\beta_v}{2}+|\alpha_3|, \frac{\beta_v}{2}+|\alpha_4|)$ (used in the computation of max_h and max_v) in order to have the target object within the view of the camera. We consider that the similarity of an object's location in the horizontal and vertical planes have the same importance (so, in our prototype the weights used for them are $\omega_{li} = \omega_{lii} = 1/2$).

4.4. Summing Up: Differences in Each Object

To compute the similarity between two images according to object o we take into account the percentage of the object visible, the viewpoint of the camera, and the location of the object in the image, which are computed as explained in the previous subsections. First, we check if object o (that appears in the image I_q) is visible in the image I; if this is not the case, then the similarity is 0. Otherwise, if the object is visible in the image I (partially or completely), the similarity between the two images regarding that object is:

$$S_{object}(o, I_q, I) = \omega_o S_{pctObj}(o, I_q, I) + \omega_v S_{views}(o, I_q, I) + \omega_l S_{location}(o, I_q, I) \quad (12)$$

where $S_{pctObj}(o, I_q, I)$ stands for the similarity according to the percentage of the object covered in the image and the percentage of the image filled by the object (see Section 4.1); $S_{views}(o, I_q, I)$ represents the similarity according to the different views of the object (see Section 4.2); and finally, $S_{location}(o, I_q, I)$ is the similarity according to the location of the object in both images (see Section 4.3). In the formula, ω_o, ω_v, and ω_l are the weights assigned to each factor and by default $\omega_o = \omega_v = \omega_l = 1/3$. So, in our proposal the three weights are equal, as there is no objective criterion to assign different weights to them. Indeed, this is completely subjective. For example, for a user two images could be more similar if they show the same percentage of the object (regardless of the percentage of the object or the kind of view obtained), and for another user two images could be more similar if they show the object in the same locations.

5. Experimental Evaluation

In this section we explain the experimental evaluation performed to evaluate our proposal. For the tests, the prototype presented in Section 3.4 was run on an Intel Core i5-480M with graphics card NVIDIA GeForce GT 540M, which is nowadays a graphics card in the mid/low-range. In addition, we recreated a scenario corresponding to a rowing boat race where there are four rowing boats moving according to the real GPS location information captured during a 20-minutes real race celebrated in San Sebastian (Spain) in 2010. In the scenario considered there are four cameras (each rowing boat has a camera aboard, on its bow) set with the same configuration as the cameras used in the real race, that is, horizontal angle-of-view 70°, vertical angle-of-view 45°, pan range $\pm 130°$, and tilt range $\pm 90°$.

In the following we describe the tests carried out to evaluate the computing time, the ranking obtained, and the user satisfaction when entering an arbitrary query image.

5.1. Evaluation of the Computing Time

In our first experiment, we want to evaluate the processing time needed by the system to compute the views of the cameras in the scenario. In this highly-dynamic environment, it is very important to process each camera view as soon as possible, as the goal is to help the TD to take better decisions in real-time. As our approach depends on the location and direction of the objects and cameras in the scenario, and in the real scenario we consider in our tests that this information is updated once every second, the system has to be able to perform the processing in less than one second to provide the TD with updated and accurate information.

In this test, we have defined an example of the most challenging query image of the scenario, where the TD wants to view all the four rowing boats (see Figure 6). This is the worst-case mentioned in Section 3.3, as the system has to generate and analyze up to ten renderings per camera when all the rowing boats are visible. For the test, we have rotated automatically each camera to view the most prominent boat of the query whenever possible. The results obtained are shown in Table 1. The minimum time needed to process the camera views was achieved in situations where some cameras did not view any boat and other cameras viewed the boats from a large distance. The maximum time needed by the system was achieved in situations where all the cameras viewed the maximum number of rowing boats possible from a close distance (notice that the maximum number of visible rowing boats for a camera on board a rowing boat is three, as

a camera cannot view its own boat). Finally, the average time for the test shows that our approach is good enough for real-time processing.

Minimum	**Maximum**	**Average**
0.148s	0.784s	0.395s

Table 1: Minimum, maximum, and average processing times of our system in a test.

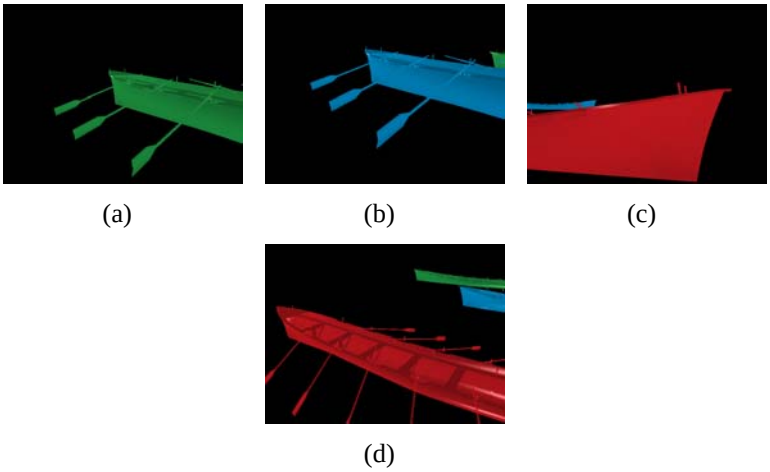
The computing time can obviously increase with the number of objects and cameras in the scenario and it is also highly dependent on the performance of the graphics card used to perform the computations. So, even though we observed a good performance in our use case scenario, in more complicated situations with a high number of cameras and/or objects it is interesting to use a better graphics card (e.g., just replacing a standard integrated graphics card with the mid/low-range GT 540M card used in the experiments divides the processing time by six, and replacing it with a mid-range NVIDIA GeForce GTX550 TI by twelve). Moreover, it is also possible to process the views in parallel on different computers, as we maintain the 3D model of the scenario updated in a database that can be accessed concurrently. These strategies will help to keep the processing time under one second even in complex scenarios, which would be needed for a smooth real-time operation.

5.2. Evaluation of the Ranking

The purpose of this experiment is to evaluate the ranked answer provided by the system for a given query image. The ranking criteria are very important because, due to the need of selecting the next camera to broadcast as quickly as possible, the TD will consider only the first positions in the ranking.

In [11] the authors emphasized that "image retrieval is only meaningful in its service to people, performance characterization must be grounded in human evaluation". However, evaluating an image retrieval system is a difficult task [12] and testing it with real users is time-consuming (so, in many approaches tests are performed with a limited number of participants [9, 8]). Moreover, in our case finding real TDs with experience in the live broadcasting of sport events able to take part in our experiments was not possible, although it would have been very interesting. So, for our test, 10 users familiarized with the use of 3D interfaces were recruited and we used four query images that could be interesting for a TD (see Figure 8). The users were presented with 45 images given in arbitrary order (see Figure 9) and the four queries, and their goal was to select the images that

they considered similar to each query and then to rank the selected images according to their similarity. To accomplish this task, the users were allowed to choose the order in which they wanted to answer the queries and modify their previous answers whenever necessary. We have selected these 45 images as they show different numbers of objects in different configurations. The number is high enough to obtain at least five similar images per query image and at the same time is low enough for the users to be able to check all the images correctly (the higher the number of images available, the higher the difficulty to keep the concentration of the user to verify them all carefully, as the amount of information that can be kept in the working memory of humans is considered to be limited [13]).



(a) (b) (c)

(d)

Figure 8: Query images used in the tests.

Considering an image as similar to another one is a very subjective matter. For some users an image is similar to a query image if it shows the same objects, for others if it shows the same percentage of the object, and for others if the viewpoint of the most prominent object is the same. So, as explained in Section 4, we used weights equally distributed when computing the similarity in our tests. According to the results obtained from the users concerning the number of similar images for each query (see Figure 10), we can observe three kinds of users: some users have a "demanding profile", as they select only a few pictures as similar (e.g., the ninth user); other users have a "lax profile", as they select more images than the average (e.g., second and eighth users); finally, the rest of users have an "average profile". Thus, we think that the system has to rank all the images without discarding any image: if we considered instead a *top-k ranking*, the appropriate k would depend

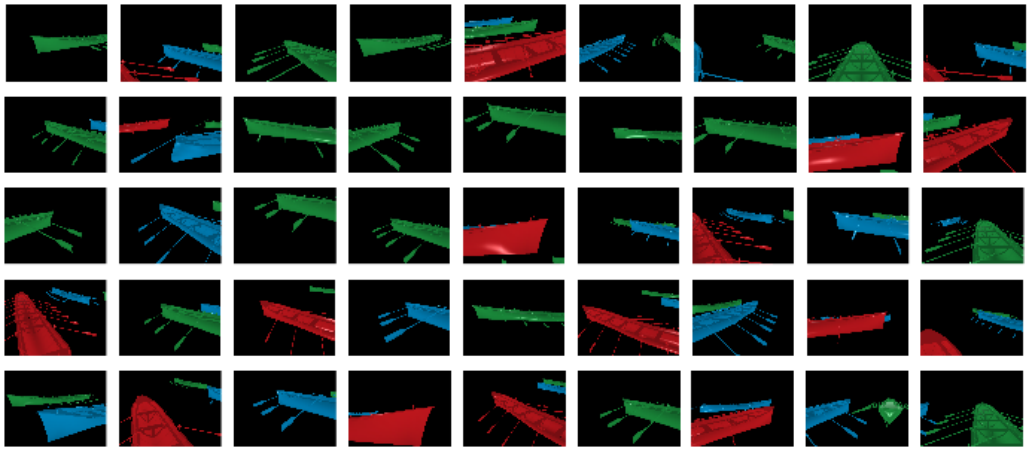


Figure 9: Set of 45 images used in the tests.

on the subjectivity of the specific user and the specific query and set of images available. However, ranking all the images also emphasizes the importance of providing a good ordering, such that images that are very different from the query image have to be placed at the end of the result list.

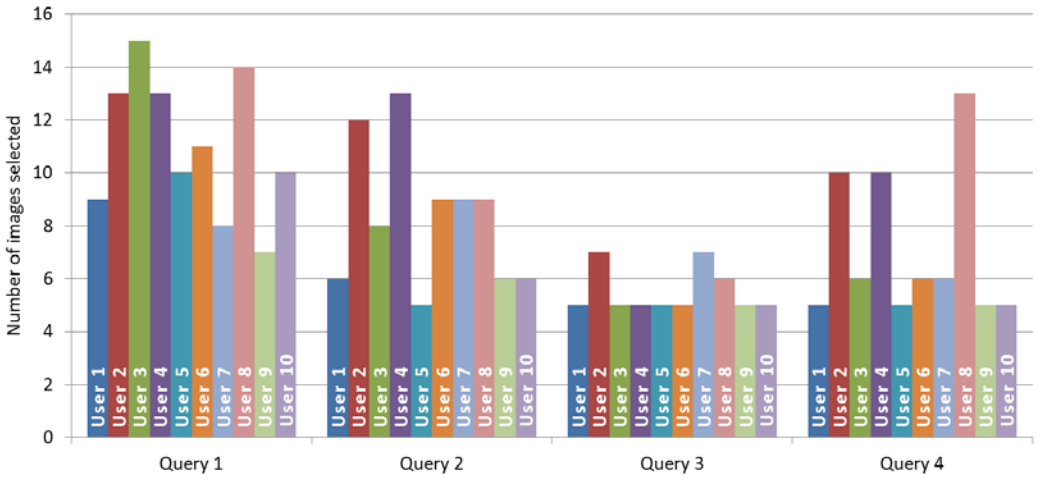


Figure 10: Number of images selected by the users as similar to each query image.

The next step is to compare the users' rankings with the ranking provided by our system. As the users' rankings for a query could include different images and a different number of results, we have to compare *partial lists of varying length*.

It is interesting to consider that it is quite common for a user to make a precise definition of the firsts positions of a ranking and the precision decreases as the element is located in the last positions. This has been confirmed by some users, that have explained that the images they placed at the end of the list had all approximately the same similarity to the query image (according to their opinion). So, we will focus on the first five images of each users's ranking (five is the maximum number of images selected for all the users for all the queries) to compare it with the system's ranking. *Kendall's tau* [14] is a measure widely used to compare rankings. However, it has the limitation that the rankings to compare have to contain the same elements (i.e., they have to be *full rankings*). In [15] a *generalized Kendall's tau* that overcomes this limitation is presented, that can be applied to compare top-k lists:

$$K^{(p)}(\tau_1,\tau_2) = \sum_{i,j \in P(\tau_1,\tau_2)} \bar{K}_{i,j}^{(p)}(\tau_1,\tau_2) \tag{13}$$

where τ_1 and τ_2 are the lists to compare and $P(\tau_1,\tau_2)$ is the union set of the elements in both lists. In addition, $\bar{K}_{i,j}^{(p)}(\tau_1,\tau_2) = 1$ if any of the following conditions hold: (a) i and j appear in both lists but in reverse order (i.e., i is ranked higher than j in one list but lower in the other); (b) i and j both appear in one list (and j is ahead i) and exactly one of i or j appears in the other list; (c) i, but not j, appears in one list, and j, but not i, appears in the other top-k list. Otherwise, $\bar{K}_{i,j}^{(p)}(\tau_1,\tau_2) = 0$, as we are considering the "optimistic approach" of Kendall's tau with $p = 0$ (i.e., $K^{(0)}$), which is a frequent instantiation of Kendall's tau in the literature. In order to normalize $K^{(0)}$ in such a way that two identical lists have a value of 1 and two lists that share no element have a value of 0, we use the normalized K [16]:

$$K = 1 - \frac{K^{(0)}(\tau_1,\tau_2)}{k^2} \tag{14}$$

We have computed the normalized distance, K, between our system's ranking and the users' ranking (the results are shown in Figure 11, where $K(u_i,s)$ represents the K value between the ranking provided by the user $user_i$ and the ranking of the system s). Values below 0.5 usually indicate that the system does not select some images that appear in the user's ranking, and values above 0.5 indicate that the system selects all the images selected by the user but the order is exactly the same only if the value of K is 1. So, for *Query 3* and *Query 4* the similarity between the rankings provided by the users and the one provided by the system

is particularly high, as with the dataset used in the experiments the users found it quite easy to select and rank images similar to those query images. However, for *Query 1* the users found more difficulties to rank the images, as there is a higher number of images that could be considered similar to the query image; in the ranking provided, for example, some users considered as more similar the images where a similar percentage of the green rowing boat was shown, whereas others considered more similar the images that showed a similar perspective.

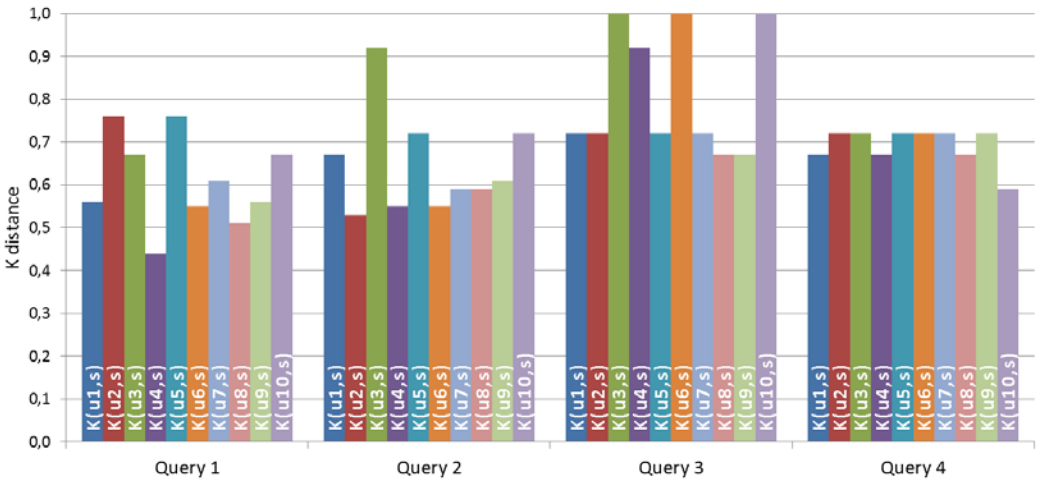


Figure 11: Comparing the ranking of images obtained by our system (s) and the users' rankings (u_i) for each query (normalized Kendall tau distance K(u_i,s)).

However, it has to be noticed that different users usually propose different rankings for the same set of images (i.e., there is some disagreement between the users about the best ranking, due to the subjectivity of the process). Therefore, comparing only the system's ranking with each of the users' rankings would be unfair. To take the subjectivity into account, we apply a similar approach to the one used in [17] to obtain the level of disagreement between the system and the users. First, we define for each query a global level of disagreement between all the users and the system, called *System Disagreement (SD)*:

$$SD = 1 - \frac{\sum_{i=1}^{M} K_{i,s}}{M} \tag{15}$$

where $K_{i,s}$ is the normalized distance K for each of the users' rankings compared with the ranking provided by our system, and M is the number of users (10 in our

case). The value of *SD* can be interpreted intuitively as follows. Considering the extreme cases, $SD = 0$ would mean that the system obtains exactly the same results (the same images in the same order) than all the users (this would be possible only if all the users provided exactly the same answer), and $SD = 1$ would mean that the system results are completely different from the results provided by any of the users. For the intermediate cases, the rankings provided by the users and the system are more similar when the value of *SD* is low.

Then, we define for each query a global level of disagreement among all the users called *Tester's Disagreement (TD)*, as the users play the role of testers for our system:

$$TD = 1 - \frac{\sum_{i=1}^{M} \frac{\sum_{j=1, j\neq i}^{M-1} K_{i,j}}{M-1}}{M} \tag{16}$$

where $K_{i,j}$ is the normalized distance K for two users' rankings. The value of TD can be interpreted similarly to what was explained before for the value of *SD*, but in this case TD measures the difference among the rankings provided by the users.

In Figure 12 we show the resulting SD and TD for each of the four queries. As $K = 1$ indicates that the two selected rankings are exactly the same, we consider that the ranking of similar pictures for a query image is correct as long as $SD \leq TD$ (i.e., when the disagreement between the users and the system is not higher than the disagreement between the users themselves). According to this, the system always behaves well except for the last query (*Query 4*), where $SD = 0.31$ and $TD = 0.27$. So, we analyze this query in the following to explain this behavior.

Figure 13(a) shows the ranking of images that the system obtains for *Query 4*. We noticed that the system locates in the fourth position an image that was not present in any of the users' top-5 rankings. This was the cause that led to obtaining a *SD* slightly greater than TD for this query. The reason for this behavior is related to the weights assigned to each term of the similarity function used by our system (Formula 12 in Section 4.4). Specifically, we consider by default $w_l = w_o = w_v = 1/3$, as there is no objective criterion to assign different weights. However, analyzing the users' rankings for the last query, we have noticed that for most of them the percentage of the objects viewed and the percentage of the shot occupied by the objects were more important than other factors. Taking this into account, we have also set the weights $w_o = 0.7$, $w_v = 0.25$, and $w_l = 0.05$, and reevaluated the query, obtaining a new ranking (see Figure 13(b)) and a new

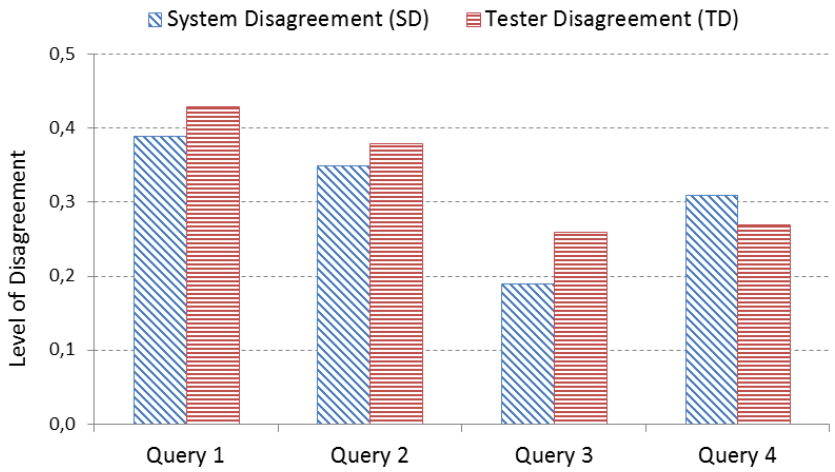


Figure 12: Tester Disagreement and System Disagreement.

$SD = 0.27$, which makes the disagreement between the system and the users equal to the disagreement between the users. So, by adapting the weights used in the similarity function (which can be performed easily by using the sliders available in the GUI, as shown in Figure 6), we can customize the system according to the preferences of a specific user. Another example of the potential interest of adjusting the weights is presented in the following section.

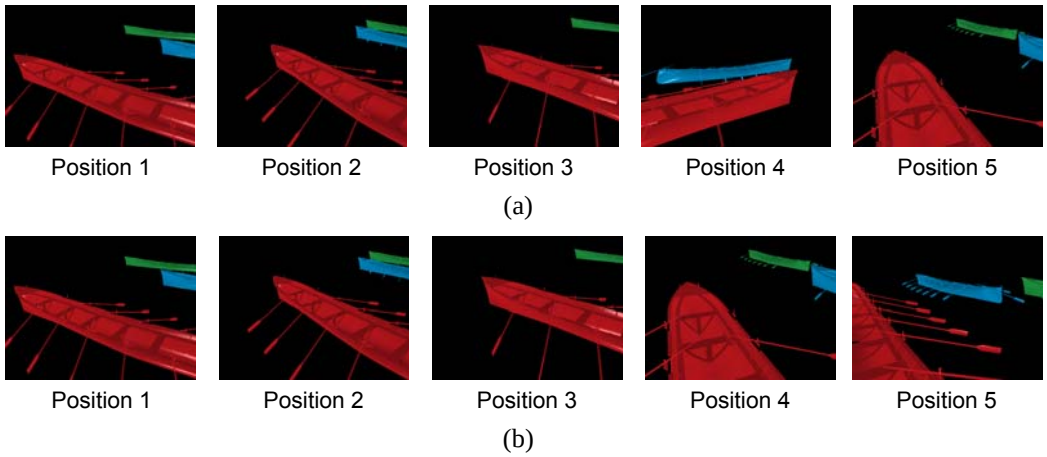


Figure 13: Ranking obtained by the system for *Query 4*: before (a) and after (b) modifying the weights of the similarity formula.

5.3. Evaluation of the User Satisfaction when Entering an Arbitrary Query

We have performed other tests where "expert" users (persons who are not only familiar with the use of 3D interfaces but also fond of photography) have used the prototype to formulate their own queries. In this section, for illustration purposes, we explain some results obtained with one of these expert users. Similar conclusions can be drawn from the tests performed with other expert users.

First, we asked the user to define a query scene; the user found no difficulties in performing this task to compose the wanted query image. Afterwards, to expand the dataset used in the previous tests, we generated some images by moving the rowing boats in the scene and by moving/rotating the camera randomly. Then, based on the new dataset of images that we generated, the system presented to the user a ranked list of images similar to his query image (see Figure 14(a)) and we asked him to make some comments about the results. He pointed out that, for him, the second image was more similar to the query than the first one, due to the viewpoint of the camera. He also noted that the rest of the images were somewhat similar to the query but he would rule out them compared to the first and second ones (the user showed a "demanding profile").

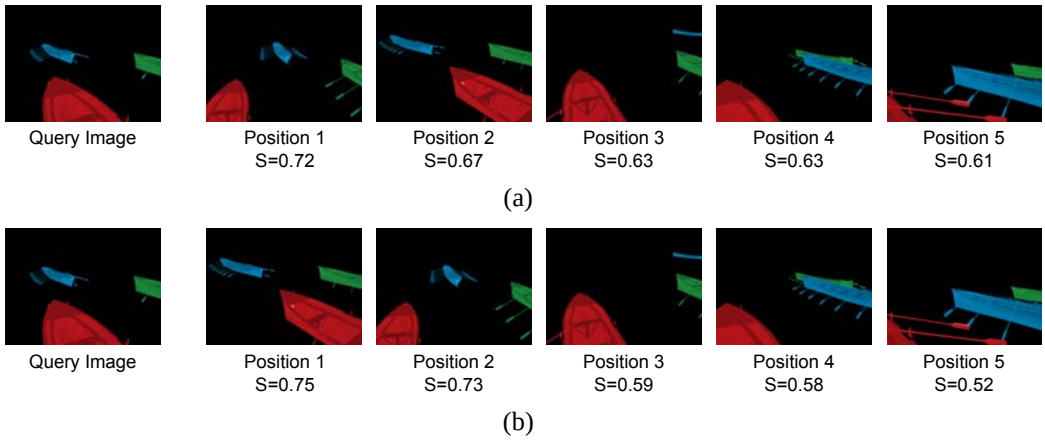


Figure 14: Ranking of images obtained for a user query: before (a) and after (b) modifying the weights of the similarity formula.

We analyzed the user answer and decided to modify the weights of the similarity function (Formula 12) to match his preferences. By increasing the weight of the views and decreasing the weight of the location (we used $w_o = 0.3$, $w_v = 0.6$, and $w_l = 0.1$) we obtained a new ranking (see Figure 14(b)) where the positions of the first two images are inverted, as the user would have expected. Moreover,

by adapting the weights to the preferences of that specific user, the differences between the computed similarity of the image in the second position and the third, fourth, and fifth increased from 0.04, 0.04, and 0.06 (in Figure 14(a)), to 0.14, 0.15, and 0.21 (in Figure 14(b)), respectively; this means that with the new weights the last three images were considered by the system much less similar to the query image than before. So, it is possible to fine-tune the weights according to the preferences of the user. Some complementary works have proposed to automatically infer suitable preference weights based on past user's interactions with the system [18].

6. Related Work

The problem of retrieving interesting images for a user has received significant research attention. Some of the proposed solutions imply the use of keywords (e.g., one of the interaction modes proposed in [19]) or a query language (e.g., [2]) to retrieve the relevant images, whereas others consider a visual input to retrieve similar images. Keyword-based interfaces for image retrieval involve several difficulties, such as the selection of appropriate keywords by the user, semantic issues (e.g., there may be several possible meanings for a given keyword, synonyms for the input keywords, etc.), and the need to annotate the existing images with the keywords that can apply to them. Among the proposals based on visual examples, which are content-based image retrieval techniques (based on the analysis of different types of visual or textual features [20, 21, 22]), we can distinguish several approaches:

- *Query-By-Icons* (e.g., [23]). The idea is to place icons that represent objects in a scene. The spatial organization of the icons is then used to retrieve images where the corresponding objects appear with a similar arrangement. This approach requires the previous annotation of the existing images with information about the identities of the objects represented.

- *Query-By-Sketch* (e.g., [8]). In this case, the user draws a sketch of the type of image he/she is interested in. A suitable user's sketching ability is required and the system must be able to deal with imprecision in the drawings.

- *Query-By-Image* (e.g., one of the interaction modes proposed in [19]). The input is now a sample image (or a set of sample images) that is used by the system to retrieve other similar images.

- *Query-By-Virtual-Example* [24]. This term was proposed in [24] to denote a solution where the input image is composed by combining the synthesis of user's gestures in front of a videocamera, a background image, and 3D object models. So, the input example is created by using virtual reality techniques. A *Support Vector Machine* (*SVM*) [25] and 3D SIFT features [26] are used to compute the similarity of images. However, the authors acknowledge that the performance of their approach is still far from satisfactory.

- *Query-By-Scene* (e.g., [9]). In this category, we could include approaches where the input is a virtual 3D scene built by the user using a 3D interface. In [9] the scene created by the user is projected onto a 2D image and then the color spatial distribution is compared with the color distribution of existing images to compute a similarity measure. So, even though we share the same basic idea –using a 3D scene to define a query image for QBI–, the image retrieval approach is different. Instead of measuring the similarity in terms of low-level features, which could lead to losing some of the semantics of the image (due to the "semantic gap"), we use high-level semantic features, extracted from the 3D scene, that enable the system to identify accurately the kind of objects and their specific identity inside a view.

Combinations of several methods are also possible. For example, in [27] a unified framework to integrate keywords and visual features for image retrieval is proposed. Some proposals advocate including additional user interaction to provide information to the system about how satisfactory the results obtained are, in order to improve future iterations for results, or to fine-tune systems' parameters [18]. As an example, [28] considers that clicks in image search logs represent implicit relevance judgments that express the user's intent and also relations between documents. A recent survey on interactive content-based image retrieval is presented in [29]. A survey on image retrieval and automatic image annotation can be found in [30].

Finally, it is interesting to mention that other proposals focus on the problem of retrieving 3D object models (e.g, [31, 32, 33, 34, 35]) or videos (e.g., [36]) instead of images. To improve the efficiency and accuracy of view-based 3D object retrieval, in [35] the authors propose to select the most interesting 2D views using a probabilistic Bayesian method (*Adaptive Views Clustering*), whereas in [33] the authors present an algorithm that minimizes the number of query views required based on information extracted from the query and the users' relevance feedback.

In [34] the query to retrieve 3D models is a set of views, but no camera constraint must be specified (so, any view set captured by any camera array can be used as a query). In [32] the authors present an algorithm to retrieve 3D models by querying-by-sketch based on the alignment of 3D models to 2D sketches. In [31] an *Interaction Metadata Query sub-Language* is presented to enable 3D model retrieval using interaction metadata. Concerning video retrieval, in [37] the authors propose the use of an ontology to find concepts related to a query.

Although there are some similarities between the proposals described in this section and the work presented in this paper, there are also significant differences. First, we focus on real-time TV broadcasting scenarios where the goal is to efficiently retrieve images provided by cameras instead of images from a preexisting image collection stored in a repository. So, both the precision and the performance of the retrieval process are key. Besides, several existing techniques (e.g., based on indexing or annotation of the images) cannot be applied due to the highly-dynamic nature of the scenario. Second, we do not apply real-image processing techniques but geometric computations based on the 3D models available: 3D representations of the real scene are automatically built by taking into account raw data such as the GPS locations of the objects and their 3D models. Identify some semantic features that our approach can consider using real-image processing techniques is too time-consuming for real-time scenarios (e.g., with our approach we know the identity of each object, and so for example we can know that a camera provides a view of the rowing boat of a certain team even if only a small portion is captured by the camera, which would be very challenging by using image processing techniques). Even though we use a 3D reconstruction of the camera views and the user query, traditional 3D matching techniques cannot be applied to our problem, as we are interested in computing the similarity considering a specific viewpoint of the 3D models (defined by the virtual camera), and so we are not interested in the similarity between the 3D models themselves. Besides, we tackle the two main research topics in the field of querying-by-example on images [24]: the development of an appropriate similarity measure and an efficient method to compute it in real-time. As far as we know, no other work focuses on retrieving images in real-time from cameras by reconstructing a virtual scenario.

7. Conclusions and Future Work

In this paper, we have presented a proposal to help a Technical Director (TD) to visually define an interesting example shot (with a certain view of one or more objects) by using an interface for the definition of 3D scenes. In this way, it en-

ables the TD to easily express the shot he/she wants to obtain. The shot defined is then analyzed by the system and evaluated over the camera sources continuously to retrieve the shots that are similar to his/her example. To this end, our system relies on a 3D model of the scene generated based on information about the interesting objects and cameras in the scenario (location, direction, and approximate 3D extent). In some situations it could be challenging to obtain this information for certain objects (e.g., it could be difficult to obtain the real-time precise location of a ball or the extent of soccer players that move their limbs while running). However, our approach does not rely on a specific technology to obtain this information nor requires a 100% precision of these data to effectively distinguish between cameras that are interesting or not for a given query. The main features of the proposal are:

- An interface for the definition of 3D scenes enables the user to define an interesting shot easily and precisely, that is used for querying-by-example.
- High-level features (such as the specific objects in a shot, their visible percentage, their viewpoint, etc.) are extracted from the example shot and the current camera views efficiently.
- A similarity measurement is presented to obtain an objective value to compare two camera shots. This value is used by the system to provide a ranking list of similar images for a query. Moreover, the system can be fine-tuned to the preferences of specific users.
- The performance of the system is suitable for its use in real-time live broadcasting scenarios. We use efficient geometric computations to compare the 3D model defining the query image with recreated 3D models that represent the views provided by the cameras.

Besides, we have developed a prototype of the system that has been used to evaluate the proposal. We have presented some experiments with users to evaluate both the ability of our system to work in real-time and to obtain results similar to those that a human could obtain. As future work, we plan to apply relevance feedback techniques [18] to automatically set appropriate weights according to the preferences of the user, inferred from his/her interactions with the system. It would also be very interesting to further test the system with real TDs in specific sport events. The evaluation with real TDs will also help us to determine their preferred way to submit queries to the system (i.e., using predefined queries – with or without parametrization– vs. ad hoc queries). Even though the use case

scenario considered in this paper is that of rowing boat races, the proposal could be applied to other sport events as long as the interesting objects are modeled in 3D and a mechanism is available to obtain the extent and location of the objects in real time.

Acknowledgments.

This research work has been supported by the CICYT project TIN2010-21387-C02-02 and DGA-FSE. We would also like to thank the anonymous reviewers for their useful comments.

References

[1] K. Choi, S. Lee, S. Y., Automatic broadcast video generation for ball sports from multiple views, in: International Workshop on Advanced Image Technology (IWAIT'09), 2009.

[2] S. Ilarri, E. Mena, A. Illarramendi, R. Yus, M. Laka, G. Marcos, A friendly location-aware system to facilitate the work of technical directors when broadcasting sport events, Mobile Information Systems 8 (1) (2012) 17–43.

[3] J. Wang, C. Xu, E. Chng, H. Lu, Q. Tian, Automatic composition of broadcast sports video, Multimedia Systems 14 (4) (2008) 179–193.

[4] R. Yus, E. Mena, J. Bernad, S. Ilarri, A. Illarramendi, Location-aware system based on a dynamic 3D model to help in live broadcasting of sport events, in: 19th ACM International Conference on Multimedia (MM 2011), ACM, 2011, pp. 1005–1008.

[5] S. Ilarri, E. Mena, A. Illarramendi, Location-dependent query processing: Where we are and where we are heading, ACM Computing Surveys 42 (3) (2010) 12:1–12:73.

[6] M. M. Zloof, Query by example, in: AFIPS National Computer Conference, AFIPS Press, 1975, pp. 431–438.

[7] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, P. Yanker, Query by image and video content: The QBIC system, Computer 28 (9) (1995) 23–32.

[8] A. D. Bimbo, P. Pala, Visual image retrieval by elastic matching of user sketches, IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (2) (1997) 121–132.

[9] J. Assfalg, A. D. Bimbo, P. Pala, Three-dimensional interfaces for querying by example in content-based image retrieval, IEEE Transactions on Visualization and Computer Graphics 8 (4) (2002) 305–318.

[10] Y. Liu, D. Zhang, G. Lu, W.-Y. Ma, A survey of content-based image retrieval with high-level semantics, Pattern Recognition 40 (1) (2007) 262–282.

[11] N. Shirahatti, K. Barnard, Evaluating image retrieval, in: Conference on Computer Vision and Pattern Recognition (CVPR'05), Vol. 1, IEEE Computer Society, 2005, pp. 955–961.

[12] B. Thomee, A picture is worth a thousand words – Content-based image retrieval techniques, Ph.D. thesis, Leiden University (Germany) (November 2010).

[13] G. A. Miller, The magical number seven, plus or minus two: Some limits on our capacity for processing information, Psychological Review 63 (2) (1956) 81–97.

[14] M. G. Kendall, A New Measure of Rank Correlation, Biometrika 30 (1/2) (1938) 81–93.

[15] R. Fagin, R. Kumar, D. Sivakumar, Comparing top k lists, in: 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'03), Society for Industrial and Applied Mathematics, 2003, pp. 28–36.

[16] F. McCown, M. L. Nelson, Agreeing to disagree: Search engines and their public interfaces, in: Seventh ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'07), ACM, 2007, pp. 309–318.

[17] R. Trillo, L. Po, S. Ilarri, S. Bergamaschi, E. Mena, Using semantic techniques to access web data, Information Systems. Special Issue on Semantic Integration of Data, Multimedia, and Services 36 (2) (2011) 117–133.

[18] Y. Rui, T. S. Huang, M. Ortega, S. Mehrotra, Relevance feedback: A power tool for interactive content-based image retrieval, IEEE Transactions on Circuits and Systems for Video Technology 8 (5) (1998) 644–655.

[19] Y. Lu, H. Zhang, L. Wenyin, C. Hu, Joint semantics and feature based image retrieval using relevance feedback, IEEE Transactions on Multimedia 5 (3) (2003) 339–347.

[20] D. C. G. Pedronette, R. da Silva Torres, R. T. Calumby, Using contextual spaces for image re-ranking and rank aggregation, Multimedia Tools and ApplicationsPublished online: 31 May 2012.

[21] Z. Shi, X. Liu, Q. Li, Q. He, Z. Shi, Extracting discriminative features for CBIR, Multimedia Tools and Applications 61 (2) (2012) 263–279.

[22] X.-Y. Wang, B.-B. Zhang, H.-Y. Yang, Content-based image retrieval by integrating color and texture features, Multimedia Tools and ApplicationsPublished online: 03 April 2012.

[23] V. N. Gudivada, V. V. Raghavan, Design and evaluation of algorithms for image retrieval by spatial similarity, ACM Transactions on Information Systems 13 (2) (1995) 115–144.

[24] K. Shirahama, K. Uehara, Query by virtual example: Video retrieval using example shots created by virtual reality techniques, in: 2011 Sixth International Conference on Image and Graphics (ICIG 2011), IEEE Computer Society, 2011, pp. 829–834.

[25] M. A. Hearst, S. Dumais, E. Osman, J. C. Platt, B. Schölkopf, Support vector machines, IEEE Intelligent Systems and their Applications 13 (4) (1998) 18–28.

[26] P. Scovanner, S. Ali, M. Shah, A 3-dimensional SIFT descriptor and its application to action recognition, in: 15th International Conference on Multimedia (MULTIMEDIA'07), ACM, 2007, pp. 357–360.

[27] E. Cheng, F. Jing, L. Zhang, A unified relevance feedback framework for web image retrieval, IEEE Transactions on Image Processing 18 (6) (2009) 1350–1357.

[28] D. Morrison, T. Tsikrika, V. Hollink, A. P. de Vries, Éric Bruno, S. Marchand-Maillet, Topic modelling of clickthrough data in image search, Multimedia Tools and ApplicationsPublished online: 16 March 2012.

[29] B. Thomee, M. S. Lew, Interactive search in image retrieval: A survey, International Journal on Multimedia Information Retrieval 1 (2) (2012) 71–86.

[30] R. Datta, D. Joshi, J. Li, J. Z. Wang, Image retrieval: Ideas, influences, and trends of the new age, ACM Computing Surveys 40 (2) (2008) 5:1–5:60.

[31] J. Chmielewski, Finding interactive 3D objects by their interaction properties, Multimedia Tools and ApplicationsPublished online: 04 June 2012.

[32] B. Li, H. Johan, Sketch-based 3D model retrieval by incorporating 2D-3D alignment, Multimedia Tools and Applications 65 (3) (2013) 363–385.

[33] Y. Gao, M. Wang, Z.-J. Zha, Q. Tian, Q. Dai, N. Zhang, Less is more: Efficient 3-D object retrieval with query view selection, IEEE Transactions on Multimedia 13 (5) (2011) 1007–1018.

[34] Y. Gao, J. Tang, R. Hong, S. Yan, Q. Dai, N. Zhang, T.-S. Chua, Camera constraint-free view-based 3-D object retrieval, IEEE Transactions on Image Processing 21 (4) (2012) 2269–2281.

[35] T. F. Ansary, M. Daoudi, J.-P. Vandeborre, A bayesian 3-D search engine using adaptive views clustering, IEEE Transactions on Multimedia 9 (1) (2007) 78–88.

[36] G. Erozel, N. K. Cicekli, I. Cicekli, Natural language querying for video databases, Information Sciences 178 (12) (2008) 2534–2552.

[37] K. Shirahama, K. Uehar, Video retrieval from few examples using ontology and rough set theory, in: 12th International Workshop of the Multimedia Metadata Community, Second Workshop on Semantic Multimedia Database Technologies (SMDT 2010), Vol. 680, CEUR Workshop Proceedings (CEUR-WS.org), 2010, pp. 5–16.