A Friendly Location-Aware System to Facilitate the Work of Technical Directors When Broadcasting Sport Events

Sergio Ilarri[1][*]   Eduardo Mena[1]   Arantza Illarramendi[2]

Roberto Yus[1]   Maider Laka[3]

Gorka Marcos[3]

[1]University of Zaragoza, Zaragoza, Spain
[2]Basque Country University, San Sebastián, Spain
[3]Vicomtech Research Center, San Sebastián, Spain

Abstract

The production costs of broadcasting sport events that require tracking moving objects are continuously increasing. Although those events are very demanded by the audience, broadcasting organizations have economical difficulties to afford them. For that reason, they are demanding the development of new professional (software and hardware) equipments that lead to a considerable reduction of the production costs.

In this paper, we present a software system that takes into account these needs. This system allows a technical director to indicate his/her interest about certain moving objects or geographic areas in run-time. The system is in charge of selecting the cameras that can provide the types of views requested on those interesting objects and areas. So, it decreases the human effort needed to produce (create, edit and distribute) audiovisual contents, giving at the same time the opportunity to increase their quality. For this, the system provides a friendly interface to specify requirements and obtain which monitoring video cameras attached to moving or static objects fulfill them, along with a query processor to handle those requests in a continuous and efficient way. We illustrate the feasibility of our system in a specific scenario using real data of a traditional rowing race in the Basque Country.

Keywords: mobile multi-camera management, location-aware systems, location-dependent queries, video broadcasting, sport events.

---

[*]Corresponding author: Sergio Ilarri, University of Zaragoza, Department of Computer Science and Systems Engineering, Edificio Ada Byron, María de Luna 1, E-50018 Zaragoza, Spain. Tel.: +34 976 76 23 40; Fax.: +34 976 76 19 14; E-mail: silarri@unizar.es.

1 Introduction

In the last years, different factors are provoking a deep revolution in the broadcast industry. First of all, there has been a strong decrease of the advertisement rates, mainly due to the economic crisis and the audience fragmentation caused by the digitalization and optimization of the spectrum (channel multiplication in satellite and terrestrial television) and the appearance of new communication platforms (e.g., video-blogs, Internet video platforms, mobile broadcasting, and so on). Secondly, continuous needs related to the upgrade of the technology (e.g., digitalization, development of media asset management systems, HD, 3D, etc.) are implying huge economic efforts. Finally, it is also evident that the expectations of the audience are increasing. The quality and richness of the content (e.g., amazing views, last generation graphics, and so on) demanded by the audience and the advertisers is much bigger than in other distribution platforms such as Internet video platforms.

In such a context, organizations focus their efforts on the enrichment and the diversification of their offers but trying to reduce their production costs. On the one hand, the enrichment is tackled by the generation of very attractive and high-quality material, including in many cases technological support (e.g., the Obama hologram in the US presidential election in 2008). On the other hand, the reduction is considered by the acquisition of new professional products that, apart from begin able to deal with the last technology (e.g., IP interoperability, HD resolution, remote control mechanisms), either have a lower price due to the inclusion of hardware coming from the consumer electronic field or add new features that decrease the human effort required for content production. An example of this is the combination of professional cameras with low-cost cameras controlled remotely. This leads to an enrichment of the content consuming experience without having a big impact on the total number of cameramen required. However, that solution has a serious impact in one of the most complex and critical tasks in a live content production environment: the more cameras are employed, the more images are available, and therefore more complicated is for technical directors (people responsible for the content production of an event) to select the best video stream to broadcast.

The audiovisual production of rowing races in the Basque Country is a relevant paradigm of this situation and it will serve us as sample scenario in this paper. The live broadcasting of such rowing races requires a very complex infrastructure: one helicopter, sailing boats with cameras and GPS transmitters, more cameras in the harbor, and a production mobile unit (usually a trailer). In such a complex context, the technical director is responsible for the selection and coordination on the fly of the graphical material, video signals, and views that are finally broadcasted to the TV audience. These tasks become especially difficult when different unexpected events happen at different geographic areas or when different moving objects become interesting at the same time.

In order to help technical directors to obtain the best broadcasting results, we present in this paper a system that helps them to select the best candidate video signals coming from static or mobile cameras (i.e., cameras installed on rowing boats, other sailing boats, fixed locations, etc.). Our proposal, based on the LOQOMOTION system [19] extended with videocamera management, relies on mobile agent technol-

2

ogy to bring the processing to the best place in the distributed wireless scenario, at any time. Thus, the camera selections provided by the system are updated continuously in an efficient manner, and it is possible to deal with different geographic areas of interest at the same time. The system can even alert about upcoming situations defined previously by the technical director (e.g., a certain object is within an area), or when some event happens (e.g., a camera gets close to a certain location). A preliminary version of our proposal appeared in [23], which has been improved and extended in this paper; among other extensions, we consider cameras that can both pan and tilt, and we analyze the precision of our system by testing a complete prototype using real GPS location data captured during a rowing race celebrated in September 2010.

The main contributions of our proposal are:

- We extend a general architecture for location-dependent query processing with videocamera management to help technical directors that broadcast sport events to deal with the multimedia information coming from different (static or moving) videocameras.

- The system enables technical directors to indicate his/her interest about certain (static or moving) objects or (fixed or moving) geographic areas predefined or defined in run-time. The system is in charge of selecting the cameras that can provide the types of views requested by the technical director on those interesting objects and areas.

- A flexible approach is followed, which can be applied to different distributed scenarios and requirements, and therefore new functionalities can be added to the system without compromising the main architecture.

The rest of the paper is structured as follows. In Section 2 we detail the features of the sample sport event used as motivating context and some interesting multimedia location-dependent queries that we would like to process automatically. In Section 3 we summarize the basic features of cameras and the work of a technical director. In Section 4 we introduce the concept of location-dependent queries, a general architecture for location-based query processing, and our proposal to extend it with the modeling and management of videocameras. In Section 5 we explain how queries in the sample scenario are processed by the proposed architecture. In Section 6 we present our prototype and some experiments that we have performed to validate our proposal. In Section 7 we review some related works. Finally, conclusions and future work are included in Section 8.

2 Motivating Context: Rowing Races in San Sebastian

Rowing racing (see Figure 1) is a very popular sport in every seaside town of fishing tradition along the north of Spain. In the Basque Country this sport is very important, and almost every town has its own team, which always counts with the unconditional local support.

Although there are multiple competitions, the most important one is celebrated in the San Sebastian bay, once a year, since 1879. The boats leave from the harbor

Figure 1: "Kaiku" boat in a race (image provided by courtesy of the Kaiku club)

where they come back after making a turn in the sea, outside the protection of the bay, covering a total distance of 3 miles. There are four lanes, although those lanes are only physically distinguished in the starting and turning points (see Figure 2).

Figure 2: Start of a race (image provided by courtesy of the EiTB Media Group)

We have chosen this race due to the fact that it is a paradigmatic example of a potential live broadcasting event that can benefit from our work. In this race, celebrated during two consecutive weekends every September, the city of San Sebastian is crowded with many visitors, and the audience of the Basque TV broadcaster is very high. The technology and equipment involved in the event have been evolving during the last years and nowadays there are multiple cameras (on sailing boats, on a helicopter, in the

harbor, on an island nearby, etc.), a GPS transmitter on every boat, a software application for the panel of judges to help them to determine the distance between the boats and the location of a boat with respect to its lane, a software tool to show on TV the real positions of the boats in a 3D reconstruction of the bay, microphones to capture the atmosphere sounds in different places, and so on.

In such a scenario, it is possible to attach a remotely controlled camera to each rowing boat. We would like to highlight that in many cases these cameras will provide innovative and interesting views from the content production perspective. For instance, these cameras may capture the exciting moment of the turning in the sea, the view of the audience perceived by the rowers, overtaking maneuvers, unusual ocean-side views of an island that is located in the middle of the bay, etc. Those views or points of interest can be classified into predefined ones (e.g., the view of spectators in the harbor) or defined while broadcasting (e.g., the view of an overtaking maneuver between two boats captured by the camera of a third boat). Figure 3 shows graphically some of the predefined points of interest on top of a 3D reconstruction of the race scenario.

Figure 3: Predefined points of interest

However, it would be crucial to help the technical director to select among the available cameras those that can provide nice views, considering that they could be remotely controlled and that they are continuously moving. It is important to emphasize that receiving the video signals is not a problem in this scenario, but the selection among the multiple video signals is a challenge. The system described in this paper helps the technical director in the identification and management of the best candidate videocameras that can view a specific (static or moving) area or object of interest; such target objects and areas can be predefined or specified during the broadcasting.

In the following, we enumerate some motivating queries that we would like to be

answered and updated continuously:

1. *Query 1: View a certain boat.* The technical director could focus on a particular boat due to many reasons (it is the local team, it is leading the race, etc.), some of them caused by unexpected situations (an accident, a broken oar, etc.) which obviously should be captured quickly from any camera.

2. *Query 2: Capture a close side view of any boat.* The technical director could want to broadcast a close side view of some boat to show the big effort performed by its team of rowers during the race.

3. *Query 3: Capture a wide view of the island from the ocean side.* This is interesting for technical directors because this view of the island is usually very spectacular (it is a sheer cliff full of seagulls) and not as typical and easy-to-access as the well-known view of the island from the bay. For this query, technical directors are interested in cameras located far from the island (to capture it completely) but within a certain range (to get also a detailed picture). So, for this query, the locations of the cameras play an important role.

Other areas that are very interesting for technical directors, and therefore could be the target of similar queries, are: the *ciaboga* area (the turning point, which is a key part of the competition), the area of the harbor or the promenade at the seaside (usually crowded with people watching the race), etc. It would be beneficial to technical directors to be able to predefine areas that are usually relevant at some time during the race. However, the system should also allow technical directors to select, while broadcasting, any area of interest by dragging the mouse on a map.

3 Technological Context: Cameras and Technical Direction

In this section, we describe the features of cameras and some basic ideas about the work of a technical director.

3.1 The Features of Cameras

Today, most of the low-cost/mid-range cameras employed in scenarios similar to the one described (e.g., F1 car races, sailing, etc.) offer a fixed view (e.g., a front view from the driver's perspective). From the production point of view, this implies important difficulties for the generation of attractive and rich content, mainly due to the limitation or lack of control of the rotation (pan and tilt) and zoom of the camera.

However, the electronics consumer sector is providing new cameras with very competitive prices that offer an acceptable image quality while providing rich remote control functionalities. Figure 4 shows some examples of these cameras, that allow a remote control of their parameters.

(a) Axis 213 PTZ   (b) Edimax IC-7000 PTn

Figure 4: Remote motorized cameras

PTZ (pan-tilt-zoom) cameras [8], and particularly those that can be controlled remotely, are used in many fields, such as corporate training, distance learning, video-conferencing, and even broadcasting. Their main specifications are the *field-of-view* (*FOV*), image quality, and remote control distance.

The field-of-view of a camera is defined based on the rotation range of both the vertical and the horizontal axis. The rotation of the axis perpendicular to the ground is called *pan* (cameras can usually pan 360° or slightly less; a left rotation is denoted by a negative value), and the vertical rotation is called *tilt* (cameras can usually tilt 180°; a downward rotation is denoted by a negative value). Figure 5 shows an example of these parameters for a robotic camera (*Sony BRC-H700*), where the pan could take values between −170° and +170° from the reference point and the tilt between −30° and 90°.

Other specifications of a camera are the *zoom* and the *focus* (they depend on the lens of the camera). The zoom is the adjustment of the focal length of the lens to make the target appear close-up or far away, and the focus is related to the clarity of the objects captured in the image. Usually the cameras have *AutoFocus*, which means that the camera is able to adjust the lens so that the target is always in focus.

This kind of robotic cameras are remotely controlled. The remotely controlled units are usually connected to the cameras (or other devices) by a serial port, for instance to an optical multiplex unit using optical fiber. Depending on the specific remotely controlled unit, it is possible to program different configurations for several cameras. These configurations can store values for the pan, tilt, zoom, and focus.

The integration of the cameras in the workflow of a live broadcasting must be supported by camera operators, who need to invest an important effort to manage the cameras and keep their concentration continually. In many cases, according to Vicomtech's experience (`http://www.vicomtech.es/`), this is a very important problem, especially when the number of cameras increases significantly. In this sense, the work presented in this paper aims to provide a tool that simplifies the management of such cameras, in order to enable the production of innovative and attractive content using

(a) BRC-H700

PAN RANGE   TILT RANGE

170°   170°

90°

30°

(b) BRC-H700 pan/tilt range

Figure 5: Rotation ranges of a camera

wireless (static or mobile) cameras controlled remotely.

3.2 The Work of the Technical Director

In television, a fundamental aspect of the job of a *technical director* (*TD*) is to manually select between the available video sources, perform live edits, and overlay text on the images when it is necessary. Figure 6 shows an example of the workspace of a TD, called *Production Control Room* (*PCR*).
There are several facilities in a PCR, such as:

- A *video monitor wall*, where multiple sources are displayed, such as cameras, video tape recorders (VTR), graphics, and other video sources.

- A *vision mixer*, which is a control panel used to select the video sources that will be broadcasted or to manipulate them (e.g., adding digital effects).

- *Intercom and interruptible feedback* (*IFB*), used to communicate with crew and camera operators.

Technical directors have overall responsibility for the operations in a PCR. They coordinate the work of the whole crew, select the video source that will be on air, and look into technical problems. TDs have to make quick decisions in live broadcast scenarios such as the one described in this paper.

Figure 6: Production-control room at the EiTB TV station

4 Monitoring Proposal

As the underlying architecture for a location-aware system for monitoring sport events, we advocate the adoption of a general-purpose location-dependent query processing system [19], that will be extended with the needed new functionalities. In order to get a better understanding of that system, in this section we present some basic concepts related to it. So, we first define and explain the concept of location-dependent query. Then, we summarize the general architecture proposed for location-dependent query processing. Finally, we indicate the new functionalities added to this system to manage references to videocameras in location-dependent queries.

4.1 Location-Dependent Queries

Location-dependent queries are queries whose answer depends on the locations of the objects involved. For example, a user with a PDA may want to locate available taxi cabs that are near him/her while he/she is walking home in a rainy day. These queries are usually considered as *continuous queries* [42], whose answer must be continuously refreshed. For example, the answer to the previous query can change immediately due to the movements of people and taxi cabs. Moreover, even if the set of taxis satisfying the query condition does not change, their locations and distances to the user could change continuously, and therefore the answer to the query must be updated with the new location data.

To express location-dependent queries, we will use an SQL-like syntax with the following structure:

SELECT   projections
FROM   sets-of-objects
WHERE   boolean-conditions
[ORDER BY   sorting-criteria]

where *projections* is the list of attributes that we want to retrieve from the selected objects, *sets-of-objects* is a list of object classes that identify the kind of objects interesting for the query, *boolean-conditions* is a boolean expression containing objects from *set-of-objects* that must satisfy the specified *location-dependent constraints*, and *sorting-criteria* is the ordering criteria that will be used for presentation of the results. The *ORDER BY* clause is optional, as in standard SQL. However, the sorting criteria can be particularly important when dealing with queries that retrieve cameras, as several cameras may satisfy the query constraints and some criteria is needed to show the most promising results first.

As an example of a location-dependent query, the query in Figure 7 asks for rowing boats that are within 0.2 miles around *boat38*. This query includes an *inside* constraint expressed with the general syntax *inside(r, obj, target)*, which retrieves the objects of a certain *target* class (such objects are called *target objects*) within a specific distance *r* (which is called the *relevant radius*) of a certain moving or static object *obj* (that is called the *reference object*). Thus, in the sample query in Figure 7 the radius of the inside constraint is 0.2 miles, and there is one reference object (*boat38*) and one target class (*RowingBoats*).

SELECT   B.id
FROM   RowingBoats AS B
WHERE   inside(0.2 miles, boat38, B)

Figure 7: Sample location-dependent query

It is important to emphasize that, although we use this SQL-like language to express the queries, higher-level facilities will be available for end users (e.g., predefined queries will be posed just by clicking on certain GUI buttons), as described in Section 6.

Moving and static objects in a scenario are not single points but have a certain geographic extension, depending on their size. Thus, objects and areas are managed in the same way (an object is characterized by a volume, which is its *extent* [38]). The techniques described in [18] to manage queries that take into account the 2D extent of the objects involved have been adapted in this paper to deal with 3D extents.

Class *Objects* is the set of entities in the scenario, which could be equipped with one or more cameras. *RowingBoats* is a subclass of *Objects*. An individual of *Objects* can be represented by the tuple:

<*id, name, extent, centroid, frontVector, topVector, cams*>

where *id* is a unique identifier of the object, *name* is the name of the object, *extent* is the volume occupied by the object, *centroid* is the centroid of the extent of the object,

10

frontVector and *topVector* are vectors pointing towards the frontal and the top part of the object, and *cams* is the list of cameras attached to the object (if any). The *frontVector* and *topVector* enable distinguishing different kinds of views of an object that could be provided by cameras (e.g., a front view of the object or a view from the top). Other interesting vectors (e.g., a vector pointing towards the bottom or the rear of the object) do not need to be defined because they can be obtained from the *frontVector* and *topVector*. For an example of the main elements characterizing a rowing boat, see Figure 8.
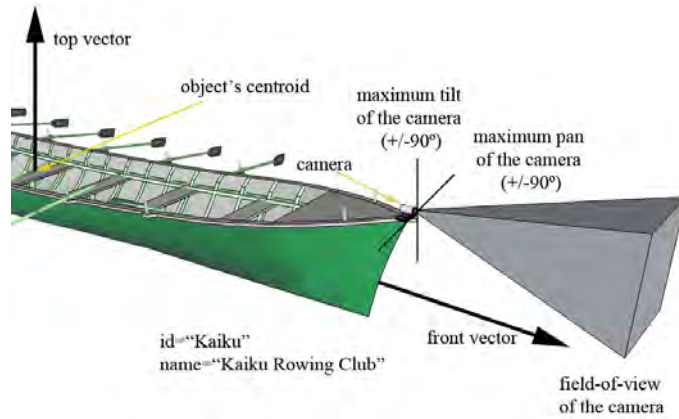

Figure 8: Main elements of a rowing boat

4.2 Processing Location-Dependent Queries

To process location-dependent queries in a mobile environment, we have proposed in previous works the system LOQOMOTION [19] (*LOcation-dependent Queries On Moving ObjecTs In mObile Networks*), a distributed location-dependent query processing system whose architecture is based on mobile agents. Mobile agents [3, 43] are programs that execute in contexts called *places*, hosted on computers or other devices, and can autonomously travel from *place* to *place* resuming their execution there. Thus, they are not bound to the computer where they were created; instead, they can move freely across different computers and devices. Mobile agents provide interesting features for distributed and wireless environments (e.g., see [40]), thanks to their autonomy, adaptability, and capability to move to remote computers.

LOQOMOTION deploys a network of agents to perform the query processing over a distributed set of objects which can detect objects moving within their range. Notice that a certain object could only detect a subset of objects in a scenario because of its limited range. The basic idea is that mobile agents move among objects in order to detect the target objects that are relevant for a query. As an example, let us assume that in the scenario shown in Figure 9 the object *Monitor* wants to retrieve the rowing boats within the area *S* centered on the black object *ref*.

In the figure, we represent with rectangles the objects that have the capability to detect other objects within its range (*coverage area*). The query will by processed
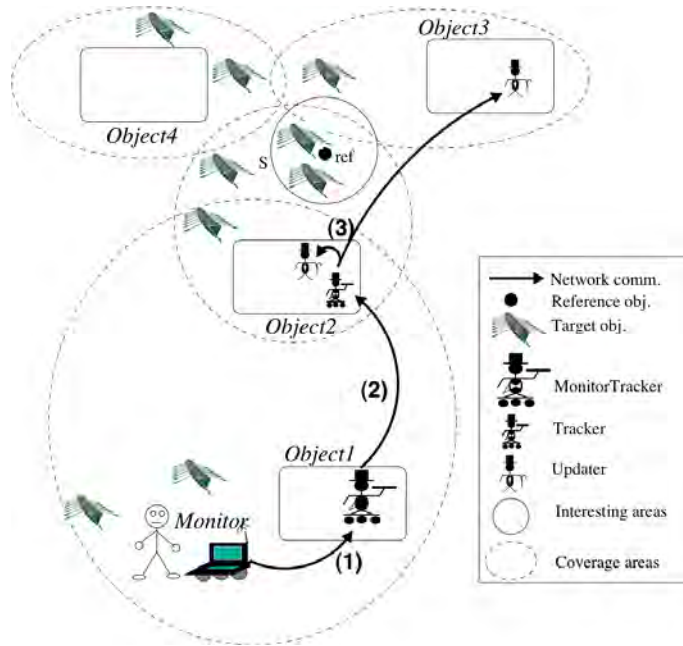
Figure 9: Architecture of LOQOMOTION

by an agent called *MonitorTracker*, that will execute on a certain object (step 1 in Figure 9). To do that, it first needs to know the location of *ref*. Notice that the object *ref* is beyond the coverage area of that object, and so the system will need to query some other object that is able to detect the *ref* object. In particular, the location of *ref* is known by *Object2* (as *ref* is within its area). So, a mobile agent called *Tracker* travels to *Object2* to retrieve the current location of *ref* (step 2 in Figure 9). Once the system knows the location of *ref*, it also knows exactly the circular area of interest *S*. Then, the rowing boats within that area are obtained, by using agents called *Updaters* (step 3 in Figure 9). Thus, one Updater executing on each object whose range intersects with *S* (in the example, *Object2* and *Object3*) will keep track of the rowing boats entering *S*. Of course, as the interesting objects move, the network of agents will re-organize itself as needed (e.g., notice that when *ref* moves the area *S* moves as well). For more details, see [19, 20].

4.3 Extension to Monitor Multimedia Data: Management of Video-cameras

In our context, one of the most interesting attributes of the objects in the scenario are obviously the videocameras, which play a key role for us. Thus, we do not only need to retrieve the objects that satisfy certain location-dependent constraints, but also filter out those objects whose cameras do not satisfy certain conditions/features needed for a suitable viewing of the target (i.e., a static or moving object or area). For this purpose,
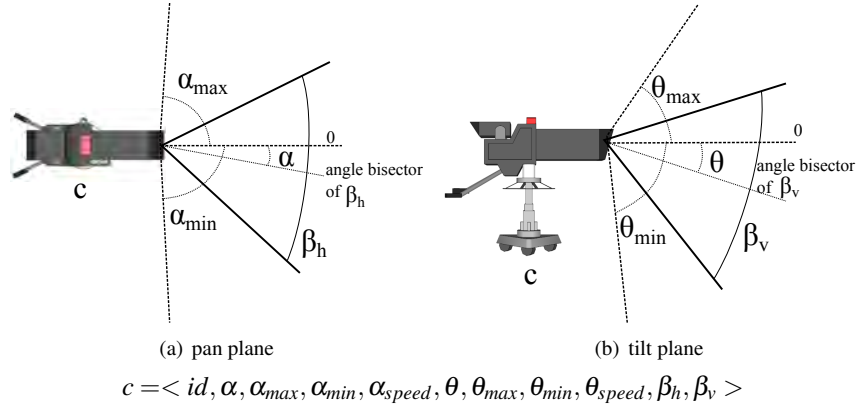
12

we model a camera c as shown in Figure 10.

(a) pan plane   (b) tilt plane

$$c = < id, \alpha, \alpha_{max}, \alpha_{min}, \alpha_{speed}, \theta, \theta_{max}, \theta_{min}, \theta_{speed}, \beta_h, \beta_v >$$

Figure 10: Modeling a videocamera

In the figure, id is a unique identifier, β_h and β_v are the horizontal and vertical angle of view, respectively. Concerning the horizontal turn (pan), α, α_{max}, α_{min}, and α_{speed} are the current pan, the maximum pan possible, the minimum pan possible, and the pan speed (degrees/second) of such a camera, respectively. Concerning the vertical turn (tilt), θ, θ_{max}, θ_{min}, and θ_{speed} are the current tilt, the maximum tilt possible, the minimum tilt possible, and the tilt speed (degrees/second), respectively. Thus, by considering both the pan and the tilt of a camera, the system is able to manage a 3D space containing 3D objects (Figure 11 shows a 3D representation of a scene).

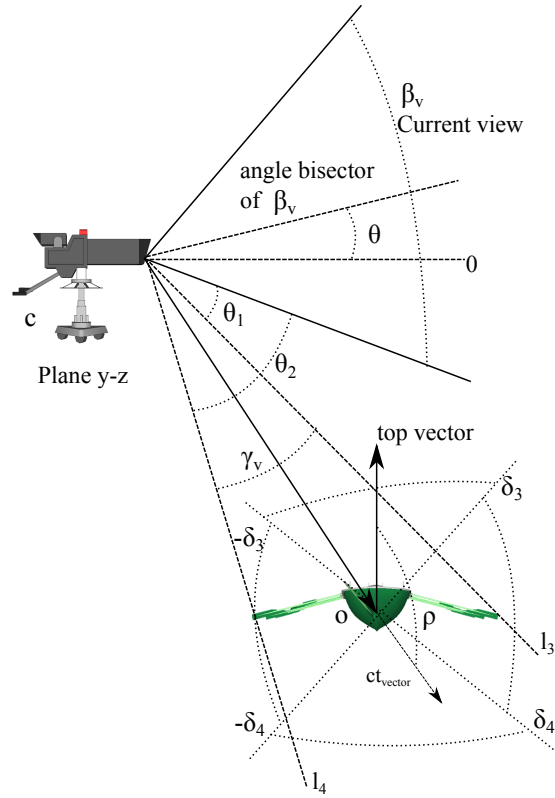Figure 11: 3D representation of a camera and a rowing boat

We define some functions that abstract us from the specific calculations needed, based on the features of the cameras, to verify certain conditions. Particularly, we define the functions *panToView* (see Figure 12) and *tiltToView* (see Figure 13).
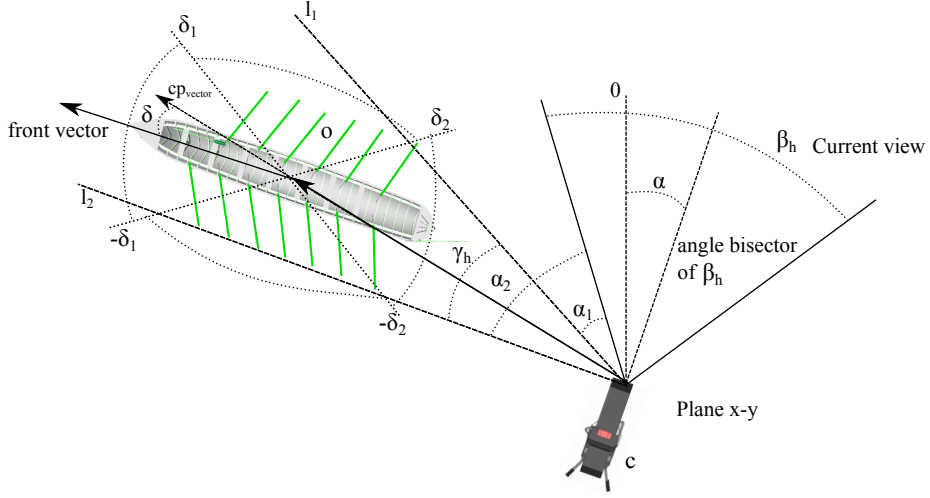
$$panToView(o,c,v,cov) = \begin{cases} \alpha_1 + \varepsilon & cov \in \{incomplete, any\} \\ & \wedge\, (view(c,o) = v \vee v = any) \\ & \wedge\, c.\alpha_{min} \leq \alpha + \alpha_1 \leq c.\alpha_{max} \\ \\ \alpha_2 & cov = full \wedge \gamma_h \leq \beta_h \\ & \wedge\, (view(c,o) = v \vee v = any) \\ & \wedge\, c.\alpha_{min} \leq \alpha + \alpha_2 \leq c.\alpha_{max} \\ \\ -999 & otherwise \end{cases}$$

Figure 12: Definition of the *panToView* function

These functions return the signed angle that the videocamera c should pan or tilt, respectively, to view an object o from a certain view v (*front*, *rear*, *side*, *top*, *bottom*, *middle*, or *any*) with the specified coverage *cov* (*full*, *incomplete*, *any*). Positive values for *panToView* mean "right pan" and negative ones mean "left pan". Similarly, for *tiltToView* positive values mean "upward tilt" and negative ones mean "downward tilt". The sign of the angles α, α_1, α_2, θ, θ_1, and θ_2, also indicate the direction of the corresponding rotation. Notice that sometimes we would need to do both movements (pan and tilt) to be able to view an object, due to the fact that the camera and the object could be at different heights. To compute the *panToView*, lines $l1$ and $l2$ (see Figure 12) are traced from the location of the camera to both sides of the geographic volume associated to the object o (in this case, a rowing boat, including its oars); similarly, lines $l3$ and $l4$ are traced to compute the *tiltToView* (see Figure 13). It is important to consider that, as explained in Section 3.1, cameras usually have mechanical limitations about

β_v
Current view

angle bisector
of β_v

θ

0

θ_1

θ_2

Plane y-z

c

top vector

γ_v

δ_3

$-\delta_3$

o ρ

l_3

ct_{vector}

$-\delta_4$

δ_4

l_4

$$
tiltToView(o,c,v,cov) = \begin{cases}
\theta_1 + \varepsilon & cov \in \{incomplete, any\} \\
& \land\ (view(c,o) = v \lor v = any) \\
& \land\ c.\theta_{min} \le \theta + \theta_1 \le c.\theta_{max} \\[2ex]
\theta_2 & cov = full \land \gamma_v \le \beta_v \\
& \land\ (view(c,o) = v \lor v = any) \\
& \land\ c.\theta_{min} \le \theta + \theta_2 \le c.\theta_{max} \\[2ex]
-999 & otherwise
\end{cases}
$$

Figure 13: Definition of the *tiltToView* function

the possible turn angles ($[\alpha_{min} - \alpha_{max}]$ for pan and $[\theta_{min} - \theta_{max}]$ for tilt). So, when the camera cannot pan or tilt to view the object *o*, satisfying the specified conditions, the above functions return a special value (in our prototype, -999). In the previous formulas for *panToView* and *tiltToView*, ε is a small value greater than 0, which must be added to the angle in order to start having the target object within the view of the camera.

The function $view(c,o)$ returns values in $\{front, rear, side, top, bottom, middle\}$ by considering in which of the quadrants defined by the angles $\pm\delta_1$, $\pm\delta_2$, $\pm\delta_3$, and $\pm\delta_4$ (the definition of front, rear, side, top, bottom, and middle views could be defined by any other partition) falls the angle between these two elements: 1) the vector defined by the location of the camera c and the centroid (of the volume) of the target object o and origin at the centroid of o (cp_{vector} in Figure 12 and ct_{vector} in Figure 13); and 2) a predefined vector which indicates the *front* or the *top* of the object o, depending on whether we are considering the horizontal dimension (pan) or the vertical dimension (tilt), respectively (see angle δ in Figure 12 and angle ρ in Figure 13). Notice that angles δ_1 and δ_2 are measured from the front vector (see Figure 12) and angles δ_3 and δ_4 are measured from the top vector (see Figure 13). Besides, angles that represent a turn to the right from the corresponding vector are considered positive and those to the left are considered negative; so, δ_1, δ_2, δ_3, and δ_4 are values that range from $0°$ to $180°$. The *view* function is defined as follows:

$$view(c,o) = \begin{cases} front & |\delta| \geq \delta_2 \\ rear & |\delta| \leq \delta_1 \\ side & (\delta_1 < |\delta| < \delta_2) \\ top & |\rho| \geq \delta_4 \\ bottom & |\rho| \leq \delta_3 \\ middle & (\delta_3 < |\delta| < \delta_4) \\ \\ & 0° \leq \delta, \delta_1, \delta_2, \rho, \delta_3, \delta_4 \leq 180° \end{cases}$$

Notice that *view(c, o)* has to be applied for both the horizontal and the vertical plane. So, according to the location of the object and the camera in Figure 12 and Figure 13, in that scenario *view(c, o)* returns "rear" and "top". Moreover, it should be noted that if a camera must pan to focus on the target it will need some time to perform the turning. We define the function *timeToPan*, which returns the time (in seconds) needed to pan the camera α degrees:

$$timeToPan(o,c,v,cov) = \frac{panToView(o,c,v,cov)}{c.\alpha_{speed}} \qquad timeToPan(c,pan) = \frac{pan}{c.\alpha_{speed}}$$

Similarly, we define the *timeToTilt* function using *tiltToView* and $c.\theta_{speed}$. The cameras can pan and tilt at the same time, and therefore the time needed to pan and tilt a camera would be $MAX(timeToPan, timeToTilt)$.

These functions will be used in the next section to take into account the features of videocameras in location-dependent queries. By defining appropriate location-dependent queries, we benefit from the power of SQL to specify in a flexible way the features of the cameras required to capture different types of views.

5 Using LOQOMOTION in the Rowing Boats Scenario

In this section, we first indicate how the different elements of the query processing architecture apply in the context of rowing races in San Sebastian:

- *Distributed Query Processing*. In the context of rowing races in San Sebastian there is a single object (a TV trailer) that is able to access all the objects (boats, people, etc.) in the scenario. Although LOQOMOTION is particularly adapted to perform well in a distributed infrastructure where there are different objects that can monitor different geographic areas, it can obviously also work when there is a single object covering the whole area of interest.

- *Inside Constraints*. As cameras that are very far from their target are usually of little interest, an *inside* constraint with an appropriate relevant radius can be used to retrieve the candidate cameras for capturing the kind of view (close, wide open, etc.) we want (in this work we do not deal with the possibility of zooming).

- *Reference Objects*. In the context of rowing races, the reference object is the interesting object (e.g., a particular boat, the island, etc.) that must be viewed. As described in Section 4.1, the extent of the objects is considered when processing the queries.

- *Target Objects*. In the context of rowing races, the target objects are the objects (e.g., boats) that have cameras that may satisfy the conditions required to view the area of interest. For example, cameras located on boats or fixed cameras in the harbor are part of the answer to location-dependent queries.

As seen above, the proposed architecture fits the context of rowing races in San Sebastian. In the rest of this section, we will analyze how the motivating queries described in Section 2 can be expressed (using an SQL-like syntax) in a way that allows their processing with the proposed architecture. For illustration, we will consider the scenario shown in Figure 14 (the boats are not done to scale, as they have been enlarged for clarity), where we assume that all the boats have a camera situated in their bow, which can pan +/-90°. We will use *o.cam* to reference the camera of an object *o*.

5.1   Query 1: View a Certain Boat

Let us suppose that we want to retrieve the cameras that can focus on a particular boat, for example the one named "Kaiku". The query would be expressed as follows:

SELECT	O.cam.id, pan, tilt, time
FROM	Objects AS O
WHERE	pan=panToView(Kaiku, O.cam, any, full)
	AND tilt=tiltToView(Kaiku, O.cam, any, full)
	AND pan<>-999 AND tilt<>-999
	AND time=MAX(timeToPan(O.cam, pan), timeToTilt(O.cam, tilt))
ORDER BY	time ASC

Besides the identifiers of the cameras, it returns the number of degrees that they should pan (*pan* value) and tilt (*tilt* value) to focus on the boat "Kaiku" (from any angle but providing a full view) as well as the time needed by that camera to do it (*time*). The cameras that currently have a full view of the "Kaiku" boat (independently of the combination of front/rear/side and top/middle/bottom that defines the view, as for example a "front and top" view of the object) appear first in the answer (*pan*=0,
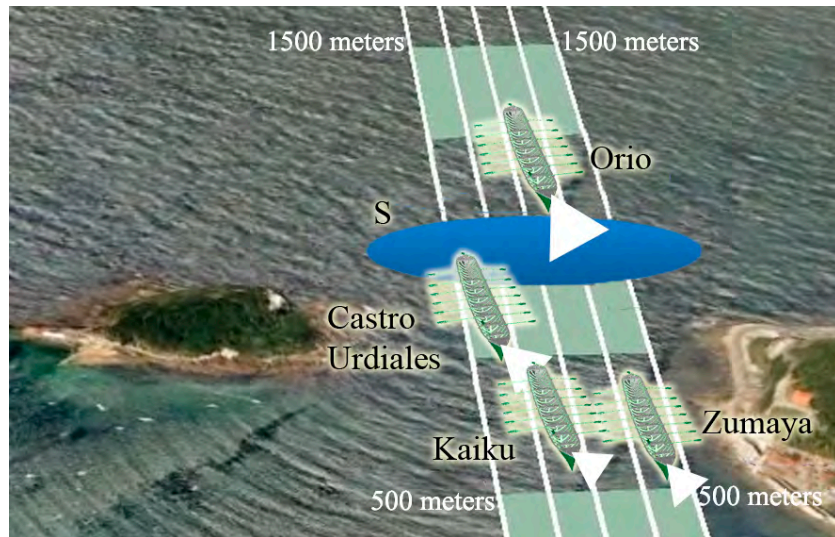
Figure 14: Sample scenario for the example queries

tilt=0, *time*=0), and the more time they need to turn (to fully view the reference object "Kaiku") the later they appear in the answer because of the *ORDER BY* clause. For example, in Figure 14 the camera of the boat "Castro Urdiales" would be retrieved first (as it has a full view of the rear of "Kaiku", see Figure 15) and the one of "Orio" would be ranked in the second position (as it can turn slightly to have a full view of the rear of "Kaiku"). Cameras that cannot view that object (e.g., the one of the boat "Zumaya" in Figure 14) are not included in the answer because of the *pan<>−999* and *tilt<>−999* constraints (see Section 4.3). However, as this query is evaluated continuously, the answer is always up to date. Thus, the information retrieved by the continuous query will help technical directors to select the best views.

5.2   Query 2: Capture a Close Side View of Any Boat

Let us imagine that we want to retrieve only cameras which are currently streaming a side view of a boat captured from a nearby location. This query would be expressed as follows:

```
   SELECT   O.cam.id, dist
     FROM   Objects AS O, RowingBoats AS B
    WHERE   pan=panToView(B.id, O.cam, side, any)
            AND tilt=tiltToView(B.id, O.cam, any, any)
            AND pan=0 AND tilt=0
            AND inside(250 meters, B.id, O) AND dist=distance(B.id, O.id)
 ORDER BY   distance
```

where we assume that a camera must be located no further than 250 meters in order to provide a close (full or incomplete) view of that boat. Moreover, we order the cameras
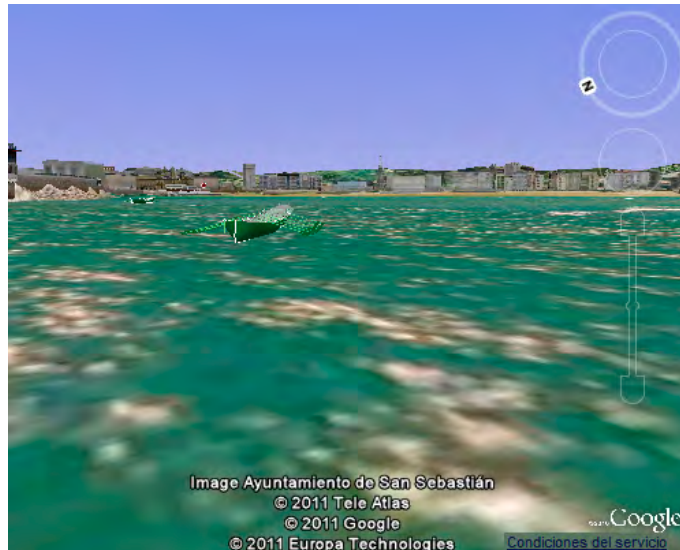
Figure 15: Virtual view of "Kaiku" provided by "Castro Urdiales" in the scenario of Figure 14

according to the distance to the target, as closer cameras are more likely to be able to provide detailed views. Thus, for example, in the scenario of Figure 14 the camera of "Castro Urdiales" would be a good candidate to view the boat "Kaiku" (see Figure 15).

5.3 Query 3: Capture a Wide View of the Island from the Ocean Side

For this query, let us assume that the technical director previously defined an area *S* (see Figure 14), on the ocean side, which represents where a camera should be located in order to have the wanted wide view of the island from the ocean side (see Figure 16). Thus, the query could be expressed as follows:

```
SELECT      O.cam.id, pan, tilt, time
  FROM      Objects AS O
 WHERE      inside(0 meters, S, O)
            AND pan=panToView(island, O.cam, any, full)
            AND tilt=tiltToView(island, O.cam, any, full)
            AND pan<>-999 AND tilt<>-999
            AND time=MAX(timeToPan(O.cam, pan), timeToTilt(O.cam, tilt))
ORDER BY    time ASC
```

where with the constraint *inside(0 meters, S, O)* we require objects no further than 0 meters from *S* (that is, objects within *S*). In the scenario shown in Figure 14 no camera is currently within the area *S*. However, the "Orio" boat will enter soon *S* and so it will become part of the answer to the continuous query. It should be noted that the technical director may want to change the size and location of the area *S* at any time, in order to

19

Figure 16: Virtual view of the island from the ocean side

focus on a different area (e.g., an accident that has happened). The system is able to adapt to changes in the monitoring requirements.

In Table 1 we summarize the current answers to the sample queries discussed in this section, according to the scenario shown in Figure 14 .

Query	Answer (cameras)
1.- View a certain boat ("Kaiku")	-"Castro Urdiales" -"Orio"
2.- Capture a close side view of any boat	For the "Kaiku" boat: -"Castro Urdiales"
3.- Capture a wide view of the island from the ocean	No answer for the moment

Table 1: Answers for the sample queries

6 Prototype and Experimental Evaluation

In order to test the system proposed, a prototype has been developed. This prototype consists of three modules: 1) a simulator, 2) a query processor, and 3) a Graphical User Interface (GUI) for technical directors. Testing this kind of system in a real-life environment is difficult because there are many real objects, devices, and wide-area scenarios involved; therefore, we have had no chance to test it on the real rowing race in San

Sebastian, as it is celebrated once a year. So, the simulator of wireless and distributed environments described in [21] has been adapted to deal with the sample scenario of this paper. This simulator used a file containing the real GPS location data of each rowing boat recorded every second during the last rowing race, in September 2010. Therefore, real trajectories were used to move objects in the simulator. Moreover, the simulator allows us to dynamically change other parameters, such as the current pan and tilt of each camera, to obtain the 3D view as a result of making a certain camera pan and/or tilt.

The technical director needs a simple GUI that simulates his/her work environment, where the query processor is integrated. As we are not using any real camera in the test, cameras are simulated using the Google Earth API (`http://code.google.com/apis/earth/`), to show an approximate view of what the cameras would view in the real scenario. This was inspired by the work presented in [28], where the data coming from GPS receivers in the boats during the broadcasting of live rowing events are integrated into a 3D terrain model. Figure 17 shows a screenshot of this GUI in our prototype. The technical director could see up to four camera streams at the same time and use the buttons on the bottom left panel to interact with the query processor, as explained in Section 5. This interface is based on mobile production units where the technical director has several input video sources to select, a screen where the source selected is processed (Preview screen), and another screen (Program screen) to see what it is being broadcasted; in our GUI, by clicking on the button "air", we "broadcast" the content of the Preview screen into the Program screen.
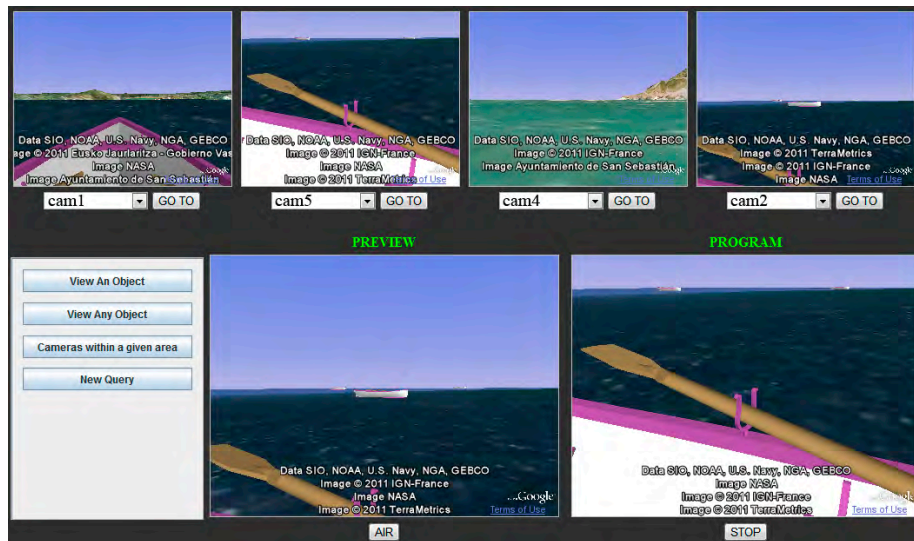

Figure 17: GUI for the technical director

The system has been tested using the query described in Section 5.1. In that query, the technical director wants to retrieve all the cameras that could provide a full coverage and any view of the rowing boat "Kaiku". The parameters used in the tests are the

following ones:

- There are four rowing boats with a camera each; they are moving according to the real GPS location information captured during the race in 2010.

- There are three fixed cameras, one on the island, another on the promenade, and the last one on a sailing boat near the rowing boats.

- The cameras are set with horizontal focus 70°, vertical focus 45°, pan range ±130°, tilt range ±90°, pan and tilt speed 5.5 degrees/second.

In the rest of this section, and we evaluate the precision and recall of the system, as well as the precision of the estimation of the time needed by a camera to focus on a target.

6.1 Testing the Precision and Recall of the Cameras in the Answer

The first test is conceived to obtain the difference between the number of cameras in an answer given by the system in real time and the optimal answer. Figure 18 shows the results of this test. We are representing the number of cameras given in the answer (which is refreshed every second). Remember that there exist seven cameras in the scenario. An error is indicated in the graph with an arrow at the specific time instant when such an error occurs. When an answer includes, for example, one extra camera (a camera that will not obtain the required view), it is denoted by "+1 Cam" and when the answer misses one valid camera (one that could obtain the required view), it is denoted by "-1 Cam". Along with the number of extra/missing cameras, we specify between brackets the amount of time (in seconds) that this error lasts.
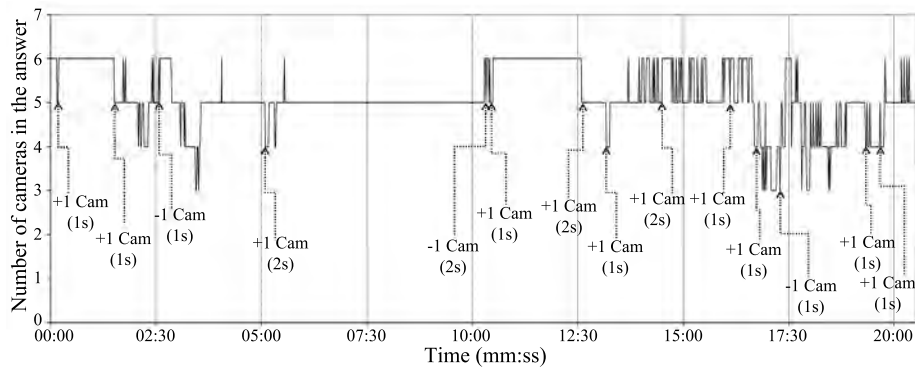

Figure 18: Error in precision/recall of the cameras

During the race, the graph shows that the coverage of "Kaiku" is quite high compared with the number of existing cameras: the maximum number is six (because the camera of "Kaiku" obviously cannot fully focus on itself) and at least three cameras

obtain the required view on the target rowing boat. Notice also that the errors committed by the system never last more than 2 seconds. The reason is the efficient continuous refreshment of the answer.

It can also be observed that the number of cameras able to capture the target varies quickly once the turning point is passed (around time=10:00). This variation is due to the query asking for a full view of "Kaiku". As the distance and speed of the rowing boats increase, some cameras are unable to maintain the target completely focused all the time (this would require exceeding the pan limit), and so they disappear for a short time from the answer set even when they can provide a view that covers most of the target object.

Even when the query processor must perform its calculations every second, the results are quite satisfactory for a period of twenty minutes, and the errors are corrected quickly enough to not be taken into account by the user, i.e., the technical director.

6.2   Testing the Precision of the Estimated Time to View

The goal of the second test is to evaluate the precision of the answer given by the query processor in terms of the time information provided. When the query processor lists a camera in the answer, the time needed by the camera to turn (pan and/or tilt) to focus on the rowing boat is computed and shown to the technical director. This estimation could be wrong because the rowing boats, and therefore their cameras, could move unexpectedly, although these errors will be corrected quickly as the query is continuously processed.

In Figure 19, we show the total error committed by all the cameras (the difference between the time given by the query processor to turn the cameras and the real time needed) at every instant. In the figure, we use triangles to mark time instants when a camera is included in the answer but its error is too big (i.e., that camera should not be included in the answer).
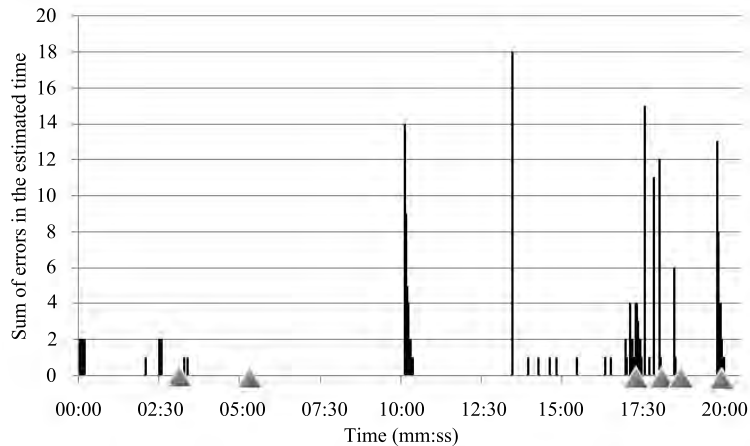

Figure 19: Error in the estimated time

23

Initially, all the cameras (pointing at the same direction as the front vector of their rowing boats) have to pan to focus on "Kaiku" and the query processor estimates the time needed for the movement with a small error. After that, the error is zero because the rowing boats are moving at approximately the same speed. Around time 02:30, the rowing boats are leaving the bay and they get closer to each other. "Kaiku" is too close to its neighbor boat whose camera onboard is not able to fully focus on "Kaiku". The next interesting time instant is around 10:00, when the rowing boats are in the turning point. Here the error is caused by the neighbor boat of "Kaiku" because the system estimates that the time to view "Kaiku" is much greater than the real time needed, due to the quick turning maneuver. Once the turning point is passed, the speed of the rowing boats and their relative distance increase. That means that the fastest rowing boats are not going to be able to focus on "Kaiku" during a few seconds, as we can see around time 14:00. Finally, the maximum distance between the rowing boats is achieved, and so there are some cameras that cannot view "Kaiku" anymore and the system incurs several errors, indicated by the triangles around time 17:30-20:00.

This test shows that the system commits small errors concerning the estimated time to view the target. Besides, those errors are fixed very quickly (in the test, in five seconds at the most), since the queries are continuously processed. The answer presented to the user contains too high errors only six times along the twenty minutes of race (remember that an answer is updated every second), and those errors only last one second.

7 Related Work

In this section, we analyze works that focus on the use of multiple cameras to produce contents (for video lectures, sport events, and others) as well as works that are closer to the field of location-based services. Works performing object tracking by using multiple cameras [24, 50] are not explicitly considered, as they pursue a different goal: estimating/monitoring the trajectory of an object from images, and in our proposal the objects are detected by using sensors [30] (specifically, GPS receivers).

7.1 Camera Management for Video Lectures

Several works in the literature have considered the problem of automatic camera management for recording and broadcasting lectures, talks and seminars [4, 5, 17, 29, 30, 31, 39, 41, 51]. In this section, we provide an overview of some proposals:

- One of the first works in the area is the *AutoAuditorium* system [4, 5] (see `http://www.autoauditorium.com/`), which uses two cameras and microphones to obtain information about what is happening on the stage and perform an automatic audio mixing, tracking of the people on stage, and automatic camera selection.

- Another interesting work is [30, 39], which also proposes a two-camera system that is able to track the presenter and the audience and generates automatically

the produced video. Several production rules have been implemented in this system, inspired by the way professional video producers work, in order to take the appropriate recording decisions. According to the tests performed, the system passes the Turing test successfully.

- The system *FlySPEC*, proposed in [29], combines a PTZ camera and a panoramic camera and integrates requests from users, along with a mechanism to solve conflicting requests. In this way, the system benefits from the involvement of the audience to reduce the probability of unsatisfactory recordings.

- As another example, [51] presents the *Microsoft Research LeeCasting System* (*MSRLCS*), that supports a scripting language to facilitate the customization of production rules for different room configurations and production styles.

- The *Virtual Director* described in [31] considers an *automation specification language* (*ASL*) that directors can use to express broadcast specifications.

Finally, it is interesting to mention the *Virtual Videography* approach described in [17]. Whereas the previous works perform a live broadcast, this last work advocates an offline processing, as this provides more time and more information to perform the video production (e.g., it is possible to look into the future).

Although the context and purpose of these works is different than the ones considered in this paper, they highlight the interest of the development of automatic video production techniques to save production costs and enable fast access to multimedia information.

7.2   Camera Management for Sport Events

Several other works have focused on helping producers of sport videos [34], as we describe in this section.

Some of these works [6, 7, 12] are part of the *APIDIS* (*Autonomous Production of Images based on Distributed and Intelligent Sensing*) project (`http://www.apidis.org/`). For example, in [7] a system that helps the video production and video summarization in the context of basketball games is presented. This system is based on a divide-and-conquer strategy and considers the trade-off among three desirable properties: *closeness* (resolution or level of detail of the images), *smoothness* (continuity in the movement and in the story-telling contents, despite the switching of camera views), and *completeness* (displaying as many objects and interesting actions as possible). Besides, user preferences are taken into account in order to provide personalized videos.

Outside the APIDIS project, the *My eDirector 2012* project, presented for example in [36], advocates extracting semantic metadata from the video streams to enable the personalization of the multimedia information broadcasted. Several problems are considered in this project, such as the detection and tracking of athletes and the automatic detection and tracking of high-level patterns. For more information, the web page of the project can be consulted (`http://www.myedirector2012.eu/`).

Besides basketball and athletic competitions, producing suitable soccer videos has also attracted the interest of research. As an example, the work presented in [14] tackles

the problem of summarizing videos (i.e., obtaining video abstracts) of soccer games by applying different image processing algorithms to analyze the input videos. Soccer videos were also considered as an evaluation scenario within the context of the APIDIS project (see [6]). The work in [2] also focuses on the analysis and summarization of soccer videos by using image recognition techniques and presents an experimental evaluation based on criteria such as the video quality, the intelligibility, the quality of the zooming and panning performed, the view size, and the view duration. As another example, a multi-camera tracking approach based on a belief propagation algorithm is proposed in [13], where thanks to the collaboration among the cameras it is possible to compensate the effects of poor observations due to phenomena such as occlusions. There are other works and patents that deal with the problem of summarizing soccer videos, such as [35].

Other sports have also been considered, such as tennis, baseball, and cricket. For example, in the *TennisSense* platform [9], nine IP cameras around the tennis court are used to automatically infer what is happening in a tennis game. The extracted metadata is then used for indexing and browsing the stored tennis videos. In [33] the problem of video summarization is considered in the context of baseball games. The proposed method is able to generate abstracts based on metadata describing the semantic content of videos encoded in the MPEG-7 format. Moreover, it takes into account the preferences of the users to build a personalized summary of the input video. Finally, in [26] excitement clips (e.g., the close-up of a player or the referee, the gathering of players, etc.) from sports videos are extracted by exploiting audio features (such as an increase in the audio level of the voice of the commentators or the cheers of the audience). The proposal is validated by using soccer and cricket videos.

All the works described in this section tackle sport events, like the proposal in this paper, but they have a different purpose. Thus, our system helps video producers to locate cameras that can provide certain images in an environment with moving objects and moving cameras, whereas the purpose of these works is usually to perform an automatic video production in an offline setting (so, for example, achieving a good performance for real-time processing is not an issue) by using image processing techniques. In other words, the focus of these works is on the operation and exploitation of the cameras, rather than on the complexity of managing multiple video signals in real time.

Although some works explicitly mention the possibility of live broadcasting, such as [47] (that focuses on soccer games and indicates that even though their system is currently performing off-line it can be improved to enable on-line processing), the context of these works is different from the one presented in this paper. Thus, for example, moving cameras are not used in those related works. However, according to [47], the automatic generation of broadcast video is facilitated when a fixed set of static cameras is considered, since the types of views available (e.g., far view, close-up view, etc.) is limited by the set of cameras used. According to [11], which presents a multi-camera selection technique based on features of objects and frames and considers a basketball game scenario (among others), an efficient mechanism for detection, tracking and feature extraction is a key element to enable view selection in a short time.

7.3 Other Works on Multi-Camera Management

In this section we briefly indicate some other interesting works for multi-camera management. An interesting and representative work concerning the management of multiple cameras and screens can be found in [10], which proposes different approaches for concurrent access and management of multiple static cameras. It is also worth mentioning the proposal in [46], where the authors work on the optimal monitoring of multiples video signals. In [1] the authors use an array of cameras to acquire a 3D scene.

However, it is important to emphasize that these representative works on multi-camera management and monitoring consider only cameras that are static (i.e., at fixed locations). On the contrary, the cameras considered in our proposal can move.

7.4 Works on Location-Dependent Queries

Developing query processing techniques that manage location-dependent queries efficiently is an intensive area of research [22, 45] , due to the great interest in Location-Based Services (LBS). The existing proposals differ in several aspects, such as the types of queries that they can process, the assumptions that they rely on, and/or the underlying infrastructure required. For example, *MobiEyes* [15] requires the cooperation of the moving objects that are the targets of the queries to process the queries distributively, *LOQOMOTION* [19] performs a distributed processing on certain objects, the proposal in [25] considers a multi-cell distributed environment to process range queries over static target objects, *DOMINO* [48] exploits some knowledge about the trajectories of the objects, *SEA-CNN* [49] focuses on the processing of continuous kNN queries, the work presented in [32] considers soft timing requirements that may exist for the processing of location-dependent queries, and [44] tackles location-dependent queries in a wireless broadcast environment through a global indexing scheme.

However, we are not aware of any work that has applied an architecture for processing location-dependent queries in the context of sport events to retrieve relevant multimedia data. Although there are works that emphasize the importance of location-dependent query processing for multimedia computing (e.g., [27]), they do not consider that the queries themselves can have as a final goal to retrieve multimedia data that are relevant for the user. Works that propose location-dependent multimedia services for mobile users (e.g., [37], where a middleware based on mobile agents is presented) are not directly related either to the research presented in this paper.

Moreover, as far as we know, the processing of location-dependent queries has not been studied before in relation to any kind of multimedia data.

8 Conclusions and Future Work

In this paper, we have shown the usefulness of location-dependent query processing in a real-world sport event: the rowing races in San Sebastian. With that purpose, we have extended a general architecture for location-dependent query processing with the capabilities needed to detect suitable (static or moving) cameras to view a particular

moving object or area. Besides, we have shown the precision of our system by testing our prototype using real GPS location data captured during a rowing race celebrated in September 2010. The main features of our proposal are:

- The features of videocameras are considered to help technical directors to access the multimedia information coming from different (static or moving) videocameras.

- Predefined or dynamically built geographic areas or static/moving objects can be set as targets for which suitable cameras can be found by the system.

- The proposal is flexible, and so new functionalities can be added to the system without important changes to the main architecture.

The underlying location-dependent query processing system used benefits from the use of mobile agents to carry out the processing tasks wherever they are needed. Thus, agents are in charge of 1) tracking the location of interesting (static or moving) objects and the field-of-view of all the cameras, and 2) refreshing the query answers continuously and efficiently.

As future work, we plan to extend the system to enable automatic tracking of targets by videocameras, in order to further help the technical directors with the viewing process. It would be interesting to extend this study to other sport scenarios too. Particularly, we think that the context of classic cycling competitions, such as the Tour of France, could be very challenging (the scenario could extend along many kilometers in mountainous landscapes where a centralized solution could be unsuitable). In other scenarios, some useful multimedia information could also be provided by mobile devices carried by people, spectators, or tourists, and so exploiting the possibilities of Mobile Ad Hoc Networks (MANETs) for content dissemination [16] and retrieval of multimedia information would be interesting. In these highly distributed and dynamic scenarios, the whole potential of LOQOMOTION could be better exploited.

Acknowledgments

References

[1] D. G. Aliaga, Y. Xu, and V. Popescu. Lag camera: A moving multi-camera array for scene-acquisition. *Journal of Virtual Reality and Broadcasting*, 3(10), December 2006.

[2] Y. Ariki, S. Kubota, and M. Kumano. Automatic production system of soccer sports video by digital camera work based on situation recognition. In *Eighth IEEE International Symposium on Multimedia (ISM'06)*, pages 851–860. IEEE Computer Society, 2006.

28

[3] F. L. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE*. Wiley, 2007.

[4] M. H. Bianchi. AutoAuditorium: A fully automatic, multi-camera system to televise auditorium presentations. In *1998 Joint DARPA/NIST Smart Spaces Technology Workshop*, 1998.

[5] M. H. Bianchi. Automatic video production of lectures using an intelligent and aware environment. In *Third International Conference on Mobile and Ubiquitous Multimedia (MUM'04)*, pages 117–123. ACM Press, 2004.

[6] F. Chen and C. D. Vleeschouwer. A resource allocation framework for summarizing team sport videos. In *16th IEEE International Conference on Image Processing (ICIP'09)*, pages 4293–4296. IEEE Computer Society, 2009.

[7] F. Chen and C. D. Vleeschouwer. Personalized production of basketball videos from multi-sensored data under limited display resolution. *Computer Vision and Image Understanding*, 114(6):667–680, June 2010.

[8] I.-H. Chen and S.-J. Wang. An efficient approach for the calibration of multiple PTZ cameras. *IEEE Transactions on Automation Science and Engineering*, 4(2):286–293, April 2007.

[9] C. O. Conaire, P. Kelly, D. Connaghan, and N. E. O'Connor. TennisSense: A platform for extracting semantic information from multi-camera tennis data. In *16th International Conference on Digital Signal Processing (DSP'09)*, pages 1062–1067. IEEE Computer Society, 2009.

[10] G. W. Daniel and M. Chen. Interaction control protocols for distributed multi-user multi-camera environments. *Systemics, Cybernetics and Informatics*, 1(5):29–38, 2003.

[11] F. Daniyal, M. Taj, and A. Cavallaro. Content and task-based view selection from multiple video streams. *Multimedia Systems*, 46(2–3):235–258, January 2010.

[12] D. Delannay, N. Danhier, and C. D. Vleeschouwer. Detection and recognition of sports(wo)men from multiple views. In *Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC'09)*. IEEE Computer Society, 2009.

[13] W. Du, J. bernard Hayet, J. Piater, and J. Verly. Collaborative multi-camera tracking of athletes in team sports. In *Workshop on Computer Vision Based Analysis in Sport Environments (CVBASE'06)*, pages 2–13, 2006.

[14] A. Ekin, A. M. Tekalp, and R. Mehrotra. Automatic soccer video analysis and summarization. *IEEE Transactions on Image Processing*, 12(7):796–807, July 2003.

[15] B. Gedik and L. Liu. MobiEyes: A distributed location monitoring service using moving location queries. *IEEE Transactions on Mobile Computing*, 5(10):1384–1402, October 2006.

[16] J. Haillot and F. Guidec. A protocol for content-based communication in disconnected mobile ad hoc networks. *Mobile Information Systems*, 6(2):123–154, April 2010.

[17] R. Heck, M. Wallick, and M. Gleicher. Virtual videography. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 3(1):4:1–4:28, February 2007.

[18] S. Ilarri, C. Bobed, and E. Mena. An approach to process continuous location-dependent queries on moving objects with support for location granules. *Journal of Systems and Software*, 84(8):1327–1350, August 2011.

[19] S. Ilarri, E. Mena, and A. Illarramendi. Location-dependent queries in mobile contexts: Distributed processing using mobile agents. *IEEE Transactions on Mobile Computing*, 5(8):1029–1043, August 2006.

[20] S. Ilarri, E. Mena, and A. Illarramendi. Using cooperative mobile agents to monitor distributed and dynamic environments. *Information Sciences*, 178(9):2105–2127, May 2008.

[21] S. Ilarri, E. Mena, and A. Illarramendi. A system based on mobile agents to test mobile computing applications. *Journal of Network and Computer Applications*, 32(4):846–865, July 2009.

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

[23] S. Ilarri, E. Mena, A. Illarramendi, and G. Marcos. A location-aware system for monitoring sport events. In *Eight International Conference on Advances in Mobile Computing & Multimedia (MoMM 2010)*, pages 305–312. ACM, Austrian Computer Society (OCG), 2010.

[24] O. Javed and M. Shah. *Automated Multi-Camera Surveillance: Algorithms and Practice*, volume 10 of *The Kluwer International Series in Video Computing*, chapter 5 "Tracking in Multiple Cameras with Disjoint Views", pages 59–84. Springer, 2008.

[25] J. Jayaputera and D. Taniar. Data retrieval for location-dependent queries in a multi-cell wireless environment. *Mobile Information Systems*, 1(2):91–108, April 2005.

[26] M. H. Kolekar. Bayesian belief network based broadcast sports video indexing. *Multimedia Tools and Applications*, 54(1):27–54, August 2011.

[27] A. Krikelis. Location-dependent multimedia computing. *IEEE Concurrency*, 7(2):13–15, April/June 1999.

[28] M. Laka, I. Garca, I. Maca, and A. Ugarte. TV sport broadcasts: Real time virtual representation in 3D terrain models. In *3DTV Conference: The True Vision – Capture, Transmission and Display of 3D Video (3DTV-Con'08)*, pages 405–408. IEEE Computer Society, 2008.

[29] Q. Liu, D. Kimber, J. Foote, L. Wilcox, and J. Boreczky. FlySPEC: a multi-user video camera system with hybrid human and automatic control. In *Tenth ACM International Conference on Multimedia (MULTIMEDIA'02)*, pages 484–492. ACM Press, 2002.

[30] Q. Liu, Y. Rui, A. Gupta, and J. J. Cadiz. Automating camera management for lecture room environments. In *SIGCHI Conference on Human Factors in Computing Systems (CHI'01)*, pages 442–449. ACM Press, 2001.

[31] E. Machnicki and L. A. Rowe. Virtual director: automating a webcast. In *Multimedia Computing and Networking*, volume 4673, pages 208–225. SPIE, 2002.

[32] Z. Mammeri, F. Morvan, A. Hameurlain, and N. Marsit. Location-dependent query processing under soft real-time constraints. *Mobile Information Systems*, 5(3):205–232, August 2009.

[33] N. Nitta, Y. Takahashi, and N. Babaguchi. Automatic personalized video abstraction for sports videos using metadata. *Multimedia Tools and Applications*, 41(1):1–25, January 2009.

[34] J. Owens. *Television Sports Production*. Focal Press, 2007. Fourth Edition.

[35] H. Pan and B. Li. Summarization of soccer video content. United States Patent 7657836, 2004.

[36] N. Papaoulakis, N. Doulamis, C. Patrikakis, J. Soldatos, A. Pnevmatikakis, and E. Protonotarios. Real-time video analysis and personalized media streaming environments for large scale athletic events. In *First ACM Workshop on Analysis and Retrieval of Events/Actions and Workflows in Video Streams (AREA'08)*, pages 105–112. ACM Press, 2008.

[37] M. H. Raza and M. A. Shibli. Mobile agent middleware for multimedia services. In *Ninth International Conference on Advanced Communication Technology (ICACT'07)*, pages 1109–1114. IEEE Computer Society, 2007.

[38] P. Rigaux, M. Scholl, and A. Voisard. *Spatial databases with application to GIS*. Morgan Kaufmann Publishers Inc., 2002.

[39] Y. Rui, L. He, A. Gupta, and Q. Liu. Building an intelligent camera management system. In *Ninth ACM International Conference on Multimedia (MULTIMEDIA'01)*, pages 2–11. ACM Press, 2001.

[40] C. Spyrou, G. Samaras, E. Pitoura, and P. Evripidou. Mobile agents for wireless computing: the convergence of wireless computational models with mobile-agent technologies. *Mobile Networks and Applications*, 9(5):517–528, October 2004.

[41] A. A. Steinmetz and M. G. Kienzle. e-Seminar lecture recording and distribution system. In *Multimedia Computing and Networking*, volume 4312, pages 25–36. SPIE, 2001.

[42] D. Terry, D. Goldberg, D. Nichols, and B. Oki. Continuous queries over append-only databases. *ACM SIGMOD Record*, 21(2):321–330, June 1992.

[43] R. Trillo, S. Ilarri, and E. Mena. Comparison and performance evaluation of mobile agent platforms. In *Third International Conference on Autonomic and Autonomous Systems (ICAS'07)*, pages 41:1–41:6. IEEE Computer Society, 2007.

[44] A. B. Waluyo, B. Srinivasan, and D. Taniar. Global indexing scheme for location-dependent queries in multi channels mobile broadcast environment. In *19th International Conference on Advanced Information Networking and Applications (AINA'05)*, pages 1011–1016. IEEE Computer Society, 2005.

[45] A. B. Waluyo, B. Srinivasan, and D. Taniar. Research on location-dependent queries in mobile databases. *International Journal of Computer Systems: Science and Engineering*, 20(2):77–93, March 2005.

[46] J. Wang. Experiential sampling for video surveillance. In *First ACM International Workshop on Video Surveillance (IWVS'03)*, pages 77–86. ACM Press, 2003.

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

[48] O. Wolfson, A. P. Sistla, S. Chamberlain, and Y. Yesha. Updating and querying databases that track mobile units. *Distributed and Parallel Databases*, 7(3):257–287, July 1999.

[49] X. Xiong, M. F. Mokbel, and W. G. Aref. SEA-CNN: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In *21st International Conference on Data Engineering (ICDE'05)*, pages 643–654. IEEE Computer Society, 2005.

[50] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):13:1–13:45, 2006.

[51] C. Zhang, Y. Rui, J. Crawford, and L.-W. He. An automated end-to-end lecture capture and broadcasting system. *ACM Transactions on Multimedia Computing, Communications and, Applications*, 4(1):6:1–6:23, January 2008.