Video-Based Sensing for Wide Deployment of Sentient Spaces

Diego López de Ipiña
Laboratory for Communications Engineering,
Cambridge University Engineering Department,
Cambridge, United Kingdom
dl231@eng.cam.ac.uk

Abstract

Sentient Spaces are perceptive physical spaces that aid user activities with computer provided services. Context information, captured through networks of sensors, is communicated to computing devices, embedded in the environment, that interpret the current situation observed and, as a consequence, trigger adequate services for the user. Unfortunately, the sensor technologies used in context capture (in special user location) are usually too expensive and complex to deploy, configure and maintain. This has prevented a wider adoption of the so-called Sentient Computing paradigm in our living or working spaces. We present, TRIP, a novel vision-based sensor that uses the combination of off-the-shelf hardware (video cameras and PCs) and printable circular markers for entity identification and location. This more convenient sensing device is accompanied by a set of programming abstractions and middleware services, named SIF, that make the development of sentient systems and their later deployment a less cumbersome and easier process.

1. Introduction

Ubiquitous Computing [27] envisions physical spaces, such as offices or homes, augmented with computing devices, seamlessly integrated into the environment, that aid humans in their everyday activities. It corresponds to the next generation of computing in which the user's attention will not be drawn by computers but vice versa, i.e. computers will be attentive to user interactions and aid their daily tasks.

Sentient Computing [8] is our approach to make the Ubiquitous Computing dream a reality. It aims to create perceptive living spaces where users activities are enhanced by software services provided by embedded devices in the environment. These environments achieve awareness of their surroundings through sensors that capture contextual information such as the location and identity of objects or the sound level and temperature of a physical space. Sentient Computing combines the dynamic information conveyed by sensors with static information from data repositories (e.g. entity attributes, the geometric features of a physical location or the capabilities of devices), in order to build an up-to-date picture of the current state of the world. Assisted by such model, sentient systems aim to perform the right service at the right time on behalf of users. In essence, computing systems and, by extension, the spaces where these systems are installed are given awareness so that they can become reactive to the people and activities taking place around them.

To be able to track the location of people and computers has been shown to be a very useful capability for the development of context-aware applications. In fact, most of the currently existing sentient applications exploit the location attribute of context [10]. This explains why in the indoors domain several positioning systems have appeared providing different entity location granularity, i.e. ranging from room-scale resolution such as the infrared-based Active Badge [25], to more accurate 3D coordinate resolution offered by systems such as the ultrasonic-based Active Bat [26]. All these systems require people and objects that wish to be located to be given an electronic tag that transmits a unique identifier via either an infrared, ultrasound or radio interface to a network of sensors on the walls or ceilings of a building. A *Location Server* then polls and analyses the information from the sensors and makes it available to applications.

Despite their proven usability, existing indoor location systems have some important drawbacks. The tags they use are proprietary and need battery power to operate. The infrastructure required, i.e. a network of sensors, is complex and expensive to install and maintain. These factors have limited the deployment of such positioning systems to research laboratories. This work presents an alternative vision-based sensor technology known as TRIP

that offers a better trade-off between the price and flexibility and the accuracy of the location data provided.

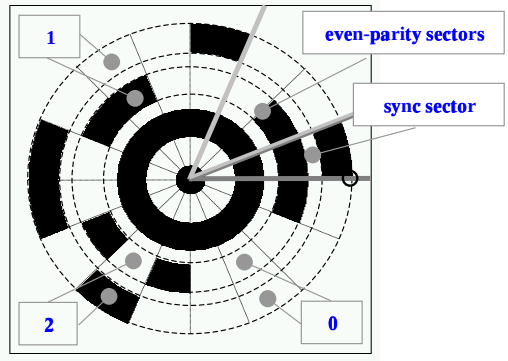
Software development for sentient systems is usually a rather involved task, since it encompasses the cooperation of several distributed elements, such as a network of sensors, a database, a Location Server and the effectors undertaking the actions triggered as result of sensor input interpretation. This appreciation has resulted in several research efforts being conducted towards facilitating context-aware applications development, e.g. the stick-e note architecture [2] and the Context Toolkit framework [21]. The SIF framework is our approach to tackle this issue. This framework differs from previous work in that it focuses not only on providing suitable high-level abstractions on which sentient application programmers base their designs, but also pays a special attention to efficient sensor data dissemination. In addition, SIF is coupled with some essential middleware services. Of special interest among these services is LocALE, a middleware infrastructure offering sentient applications control over object's lifecycle and location in a network.

Sections 2 and 3 explore the TRIP technology and the distributed systems infrastructure built on top of it. SIF is overviewed in section 4. Section 5 offers a brief description of the LocALE middleware service. Section 6 illustrates the usability of the TRIP technology through the description of two TRIP-enabled applications built with the assistance of the SIF-defined programming abstractions and the LocALE middleware. Section 7 offers a brief summary of some related research. Section 8 sketches some further work and draws some conclusions.

2. TRIP: a vision-based location sensor

TRIP (Target Recognition using Image Processing) [12][13] is a vision-based sensor system that uses a combination of visual markers (2-D *ringcodes*), see Figure 1, and video cameras to identify and locate tagged objects in the field of view. Relatively low CPU demanding image processing and computer vision algorithms are applied to video frames to obtain, in near real-time, the identifier (*TRIPcode*) and *pose* (location and orientation) of the targets with regard to the viewing cameras.

TRIP constitutes a very cheap and versatile sensor technology. Its 2-D printable ringcodes (*TRIPtags*) represent a ternary number in the range 1-3^{13} (1,594,323). This number is read anti-clockwise from its unique synchronisation sector, as illustrated in Figure 1. Only off-the-shelf hardware is required: low-resolution CCD cameras and the processing power of PCs. TRIP's low-cost and ample addressing space make it suitable for tagging even low-cost items, such as books or office stationary, in contrast to other location sensing technologies. Users with web-cams attached to their PCs may install the TRIP software and hence provide visual awareness to their machines. A public domain release of the TRIP software is planned to be available soon at our research group's site: http://www-lce.eng.cam.ac.uk.



* 10 2011221210001

Figure 1. TRIPcode representing 1,160,407

2.1. TRIP target recognition process

The TRIP software implements a target recognition algorithm that receives as input raw video frames and executes a set of image processing stages in order to determine the identifier and location of the centre of a TRIPtag in a video frame where TRIPcodes have been spotted. Below an overview of this process is given. For more details refer to [13].

Figure 2 depicts the video filtering process undertaken by the target recognition algorithm. Firstly, the original image is transformed into a binary image (containing only black and white intensity values) with increased contrast level, by means of an adaptive thresholding process. This stage makes TRIP very robust to variable lighting conditions and suitable even for low-quality CCTV or web cameras. Secondly, the pixels where the intensity values undergo a sharp variation (edges) are identified, by applying the gradient operator. Next, edges are thinned to 1-pixel width. Fourthly, the connected chains of edge points are followed producing for each edge tracked a list of ordered point locations. Note that as result of the camera's perspective effect targets' circular borders are imaged as ellipses. Thus, in this stage only edges whose shape is likely to define an elliptical shape are kept. In fifth place, an ellipse fitting procedure is applied to each edge encountered, obtaining the ellipse parameters best approximating the edge points. Sixthly, the ellipses found are tested for concentricity in order to identify candidate TRIPcodes. Then, a code deciphering stage is applied based on the outcomes of stages 1 and 5. As part of this stage the code obtained is validated through an even parity error check.

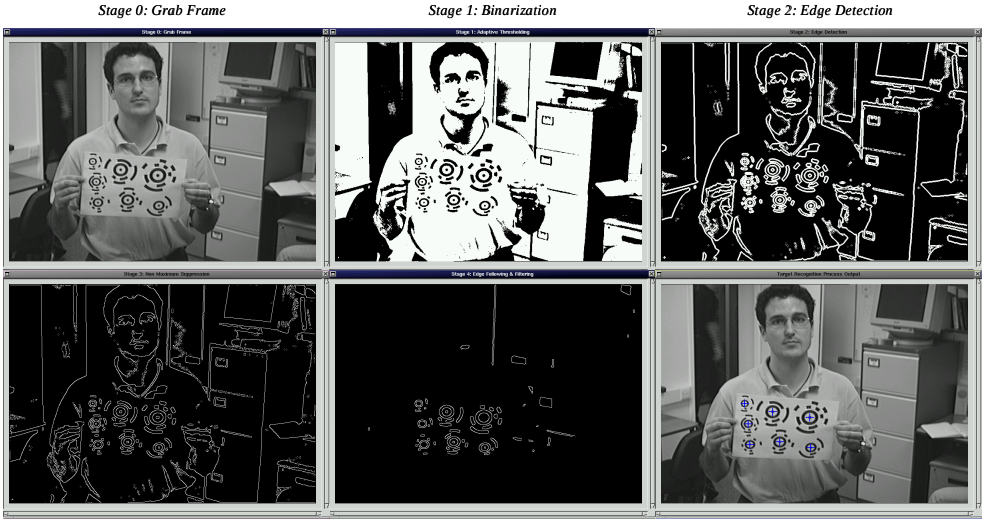


Figure 2. TRIPcode Recognition Algorithm

Finally, the identifier and geometric properties of the outermost elliptical border of each target are returned.

2.2. TRIP target pose extraction

The target pose extraction algorithm determines from a single view of a TRIPtag the orientation of the target plane and the position of its centre with regard to the viewing camera. It takes as input the geometric features of the elliptical borders of the spotted targets returned by the target recognition algorithm. For each TRIPcode identified, it establishes a *homography* or transformation that back-projects the elliptical outermost border of a target, as seen in the image plane, into its actual circular form, of known radius, in the target plane. The size of a target may be encoded within the 13 ternary bits reserved for identifier encoding in a TRIPcode, or otherwise passed as a parameter to the parsing process.

TRIP makes the assumption that the camera intrinsic parameters are known. In other words, the user is required to undertake the well-known camera calibration [24] process once for each CCD camera. Fortunately, there are several publicly available software packages for camera calibration, e.g. [24].

The target pose extraction algorithm returns the translation vector (t_x, t_y, t_z) and rotation angles (α, β, γ) that define the rigid body transformation between the camera coordinate system and a target centred coordinate system. This method exploits the property, unique to a conic that is a circle, that the back-projected curve's implicit equation must have equal coefficients X^2 and Y^2 and no term in XY. It is based on the POSE_FROM_CIRCLE algorithm described in [5].

The projective geometry involved in the algorithm is shown in Figure 3. The image ellipse of the outermost circular border of a target's bull's-eye defines a cone with vertex in the centre of projection of the pinhole camera (O). The orientation of the circle's plane, π_t, is found by rotating the camera so that the intersection of the cone with the image plane becomes a circle, which happens when the image plane is parallel to the target plane. This rotation is estimated as the composition of two successive rotations. The first puts the Z-axis through the centre of the target, and aligns the X and Y axes with the axes of the image ellipse; the second rotates the newly found coordinate system around Y' until the image plane, π_i, becomes parallel to π_t. The rotation of the target's outermost circle around the Z-axis orthogonal to its plane, impossible to recover from the view of a circle given its symmetry, is obtained by using the bottom outermost corner of a TRIP target's synchronisation sector (denoted by a small circle in Figure 1). The projective geometry involved in this process is fully described in [12].

2.3. TRIP Sensor Operation Modes

TRIP's image parsing process, despite being optimised to require the least possible processing load, still has relatively high processing demands. Nevertheless, a TRIP sensor will be often analysing frames where no TRIPtags are encountered, or alternatively, frames where the approximate location of a TRIPtag in the image could be inferred based on its location in previous frames. Therefore, it is convenient to provide the TRIP sensor with some intelligence both to distinguish the worthiness of

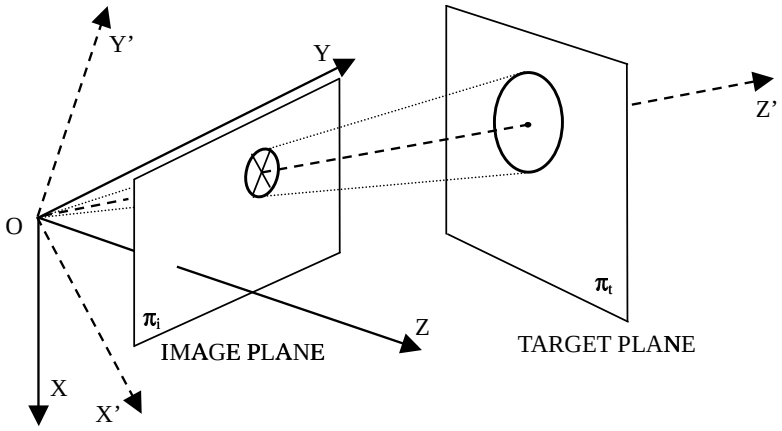


Figure 3. POSE_FROM_CIRCLE geometry

analysing an image, and if the analysis proceeds, to determine whether the full-image parsing should take place or this process should just be applied to a smaller sub-window in the image. The aim is to give to TRIP maximum responsiveness but still guaranteeing the minimum consumption of processing load. This is the reason why TRIP presents an adaptive behaviour during its operation. The sensor transits through 3 different operating modes, named *default, saving* and *real-time* modes, depending on the characteristics of the scene viewed. Figure 4 shows a state diagram with TRIP's operation modes and the conditions upon which the sensor switches from one behaviour mode to another.

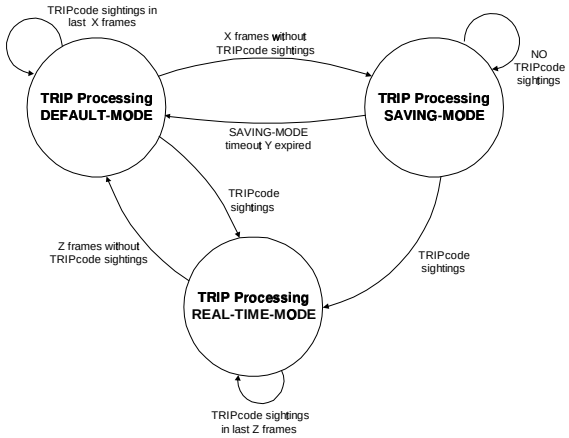


Figure 4. TRIP sensor's operating modes diagram

The TRIP *default-mode* of operation is activated on sensor start-up and every time a timeout set to guarantee full-image analysis expires. This timeout, specified as a parameter in the system bootstrap, limits the time the sensor spends in either the saving or real-time modes, where a full-image analysis may not have taken place. This is necessary to enforce a good responsiveness of the system to scene changes.

The TRIP *saving-mode* of operation is triggered by the default-mode whenever there are no TRIPcode sightings in a given number of frames specified at the system start-up. The distinctive feature of this mode is that it introduces a *Triggering Analysis* stage before the standard TRIP parsing procedure to an image is undertaken. This stage determines whether significant changes in the scene have occurred that make worthy a full image analysis. Every given time slot the triggering module analyses a low-resolution version of a newly captured image (i.e. one every 3 pixels in both horizontal and vertical directions) and compares each pixel with a previously calculated Running Video Average (RVA) frame of the background. An RVA [22] is a method to construct an evolving background frame, insensible to small variations in day illumination, using the average pixel values of the N preceding images. The triggering stage determines the percentage P of pixels that have changed by more than a certain threshold. If that percentage is high a full-image analysis is triggered, otherwise the TRIP sensor is set to sleep.

The TRIP *real-time-mode* of operation is triggered from any of the other two modes every time TRIPcodes are spotted within an image. This method exploits the spatial locality of TRIPtag images in successive video frames. Once a target has been identified within an image, subsequent frames are only explored within a small pixel window containing the area on the previous image where a target was recognised. Nevertheless, in order to guarantee the rapidly identification of newly appearing TRIPcodes, the full image processing is still carried out frequently, by default unless otherwise specified once every 5 frames.

2.4. TRIP performance and accuracy

The current C++ implementation of the TRIP Target Recognition algorithm processes 15 640x480 pixels frames per second on an 800 MHz Pentium III. When the target recognition and pose estimation are simultaneously undertaken, the performance achieved on the same machine is about 12Hz. Near real-time video processing is achieved through the adaptive behaviour of the TRIP sensor's operation.

TRIPtags are recognised as long as the slant between the normal to the target plane and the translation vector is less than 70° and the target image occupies an area of at least 20x20 pixels. Targets spotted in a frame of 640x480 pixels resolution are identified as long as they are within 3 meters distance. The pose extraction method returns the 3-D location of the centre of a TRIP target with regard to a viewing camera with an average error of less than 3%. The likelihood of finding false positives is negligible thanks to the error parity check used in the TRIPcode encoding.

3. TRIP: a distributed location sensor

An event-based distributed architecture has been devised around TRIP in order to export the sensor data provided by this technology to interested applications. The functionality of the TRIP system, i.e. its target recognition and pose extraction, has been encapsulated within a CORBA [19] component named *TRIParser*. This component offers a UNIX `pipe`-like interface that enables applications to connect distributed *Frame Source* components, supplying images from cameras spread throughout the environment, to TRIParsers. A TRIParser may pull images from one or more Frame Sources. Every frame pulled is tagged with a unique camera identifier. TRIP processing results are, by default, asynchronously communicated in event-form to a CORBA Notification Service's Channel [18] associated with each TRIParser. This interleaved event communication component decouples analysers' frame processing and result reporting duties. Thus, TRIP can concentrate on the CPU intensive image parsing process whereas the Notification Channel component deals with supplier and consumer registration, timely and reliable event delivery to registered consumers, and the handling of errors associated with unresponsive consumers.

A TRIParser generates *TRIPevents*, represented in pseudo-C++ code in Figure 5, that are mapped into the *Structured Event* message type supported by the CORBA Notification Service [18]. Structured Events define a standard format for messages conveyed to a Notification Channel. In the body of this message the contents of an event are mapped into a set of name-value pairs to which filtering operations can be applied. Parties interested in a given TRIParser's raw location data subscribe to its channel passing a set of constraints over those name-value pairs of an event, expressed in the Extended Trader Constraint Language [16]. The Notification Channel performs event filtering and communication on behalf of its representing TRIParser. Hierarchical interconnections of TRIP Parsers' Notification Channels can be created in order to ensure the efficient and scalable dissemination of TRIP generated sensorial data. For example all the Notification Channels corresponding to TRIParsers in a room could be federated, in order to make available the whereabouts of TRIPtag wearers within a room.

```
struct TRIPsighting {
 string TRIPcode; // code ternary representation
 paramsEllipse params; // bull's-eye's outer ellipse
                       // params (x,y,a,b, θ) in image
 double d2Target;
 targetPosition position;  // (xpos, ypos, zpos) vector
                       // from camera to target origin
 targetOrientation orientation; // (α, β, γ)
};
```

```
struct TRIPevent {
 unsigned long cameraID;
 TimeStamp time;  // secs and usecs since 1/1/70
 TRIPsightingList sightingList;
};
```

Figure 5. TRIPevent contents

A TRIParser also provides a synchronous invocation interface (`parseFrame`) by which the TRIParser pulls a frame from the source passed as a parameter and returns the location data inferred from it. Hence, applications can interact with a TRIParser in either a synchronous or asynchronous form.

3.1. The TRIP Directory Service

A TRIP Directory Server (TDS) has been created with the purpose of regulating the TRIPcode granting process and establishing mappings between real-world objects and TRIPcodes. This component guarantees the efficient utilisation of TRIPcodes' addressing space and their classification into categories with a common ternary prefix, used by consumers in event filter registration. The TDS offers CORBA interfaces for the creation, modification, deletion and retrieval of both TRIPcodes and their categories. On category creation, a data schema defining the set of name-value type pairs associated to TRIPcodes belonging to that category is specified.

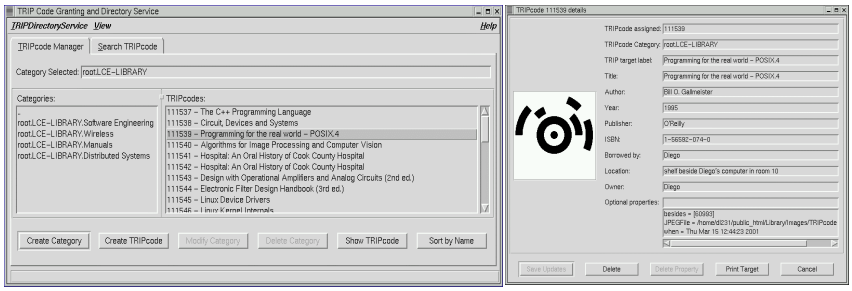


Figure 6. TRIP Directory Service's GUI front-end snapshot

The TDS also provides a notification mechanism suitable for applications that only want to query this component during their bootstrap stage and still be aware of modifications to their TRIPcode's categories of interest. TRIPcode creation, modification and deletion events are conveyed to the TDS's associated channel. Figure 6 shows a snapshot of the TDS's GUI front-end. This application enables the user to easily input TRIPcode associated data and to print out their respective TRIPtags.

4. The Sentient Information Framework

The *Sentient Information Framework* (SIF) defines an application construction model to streamline sentient systems development. SIF isolates context capture and abstraction from its use by applications and, at the same time, it provides efficient mechanisms for context communication. Its main function is to transform context information into the format demanded by applications, rather than enforcing them to deal with the low-level intricacies of sensor access and data interpretation. SIF-enabled applications simply subscribe for the high-level contextual event notifications that can directly drive their operation. For example, a "door access control system", conventionally, would have to interact directly with the underlying identification technology, e.g. TRIP, and interpret the sentient data provided, to determine when an authorised user presence is detected. Under SIF, however, this application would simply register for an "authorised person presence" event, since SIF undertakes the sentient data manipulation and interpretation on its behalf.

The architecture of SIF (see Figure 7) consists of a group of co-operating distributed software components that use events as a uniform way of informing each other of context changes. SIF defines three main component *abstractions* upon which the design of sentient applications may be based:

1. *Context Channel* (CC) objects interleave in between other SIF components to decouple their operation from event filtering and communication duties. Through them, multiple suppliers can communicate transparently and asynchronously with multiple consumers without the components knowing about each other. Physically implemented as OMG Notification Channels, they are shared by co-operating components that either generate or consume events of the same type. Upon event arrival they undertake event filtering and transmit to consumers only those events for which they registered interest.

2. *Context Generators* (CG) encapsulate a single context source or a set of related ones and the software that acquires raw context information from them. Every CG communicates the sentient data captured to its associated CCs. CGs may be based in physical sensors such as TRIP, GPS, or a simple microphone. They can also represent a software object, such as the TRIP Directory Server, that communicates notifications about changes in its state.

3. *Context Abstractors* (CA) achieve the separation of concerns between context sensing and application semantics. They consume raw sentient data provided by CGs (e.g. TRIPcode 1234 spotted), interpret its contents (e.g. user Diego spotted) and augment it (e.g. Diego's login is dipina), producing enhanced contextual events that directly drive applications. Sometimes they also undertake event aggregation, i.e. a set of atomic events coming from other CAs or CGs is summarised by a single new event.

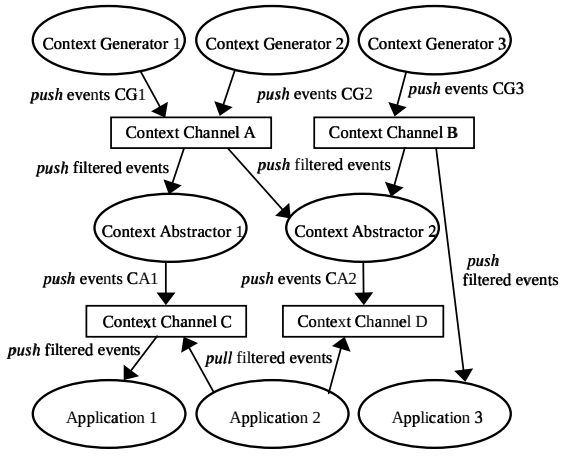


Figure 7. The Sentient Framework Architecture

Both CAs and CGs are conceived for asynchronous and decoupled interaction with other SIF components by means of their associated CCs. Nevertheless, they also provide a uniform synchronous interface to enable applications to query the latest context they have monitored. Querying a CA or CG is appropriate for one-time context needs or when the application is first started. The notification or publish/subscribe mechanism is appropriate for repetitive context needs.

An interesting side effect of having implemented CCs with CORBA Notification Channels is that event suppliers are offered a *quench* mechanism by which they can dynamically realise whether there are consumers interested on the events they are producing or not. This facility permits context sources to intelligently stop or resume event communication, hence making an efficient use of the network.

CA components are the core components of the SIF architecture. They undertake the separation of context capture and interpretation from context use by transforming the context information flow to the end application needs. Their behaviour responds to an Event-Condition-Action (ECA) model. They monitor for asynchronous event communication, apply conditional statements over them and whenever a condition is fulfilled, generate an action (in this case, produce a higher level event). Between a CG and a final application, an event can flow through N-tiers of CA components. For example, a person's TRIPtag sighting event, "code 1223 seen by camera 30", could be transformed by a *Location Service Context Abstractor* into "Diego seen at entrance corridor's front-door region". In turn, a *Lab Access Context Abstractor* could map this latter event into an "authorised person presence" event. This enhanced contextual event would finally reach the application in charge of performing the actual action, i.e. the door opening effector.

An essential middleware service, named *Context Trader*, is used to assist SIF components and applications in the discovery of other SIF components and their capabilities. Its operation is analogous to a conventional Trading Service [16]. When a SIF component is started it exports to the Context Trader its object reference and the metadata associated to the types of events and the values of the event attributes it can provide. In addition, it also reports metadata about the set of services it can provide, e.g. frame grabbing capabilities. Applications may later request through constraint expressions for the object references of the SIF components providing a given type of contextual event, or whose events contain the set of attribute name values specified. Furthermore, applications may register with the Context Trader to be notified when new context sources providing their events of interest are added or when existing ones are unregistered. This permits sentient applications to adapt to changes in the SIF framework.

The main benefits of adopting SIF for sentient application construction are its run-time *reusability* and *extensibility* features. CCs are shared by context sources (CGs and CAs) producing events with previously agreed semantics and by consumers (CAs and applications) interested on that contextual data. This property can be useful for Sensor Fusion purposes. New CAs can be interposed in between other CAs or CGs and applications in order to adapt and extend the contextual information flowing through SIF to the end application needs. Our context abstraction mechanism permits an application to receive context from an alternative sensor system to the initially used one without actually requiring any change in the application code.

5. Middleware for object lifecycle control

Thus far, we have supported our aim of facilitating a wider deployment of sentient spaces with two contributions: TRIP and SIF. TRIP provides a cost-effective and easily deployable location sensor technology. SIF efficiently manipulates and disseminates contextual data, simplifying application development. One question still to be answered, however, is how, once a sentient system receives a high level event (e.g. "Diego enters his office"), can the system effectively trigger an action, i.e. activate, deactivate or migrate a user-bound software service?

Experimentation with sentient application development has shown the need for an infrastructure to ease user-associated services' lifecycle control. Otherwise, every time a new sentient application is developed, the distributed components involved in its operation have to be manually started, or at least the factory objects capable of creating these components on demand. The lack of this middleware makes sentient application deployment very cumbersome. Moreover, user-related services are often bound to his location and must follow the user as he moves through the physical space. Therefore, it is desirable to control both the lifecycle of user-bound services and their location in a host network.

The LocALE (Location-Aware Lifecycle Environment) framework defines a simple mechanism for managing the lifecycle (i.e. activation, migration and deactivation) and location of distributed CORBA objects residing in a network. In addition, it adds load-balancing and fault-tolerance features to the objects whose lifecycle it controls. The emphasis of its design is placed on providing a suitable interface for third party *object-location controllers*. These controllers, aware of personnel location and computing resources location, load or capabilities, can intelligently direct components' locations and lifecycles.

LocALE is limited to offering the object-lifecycle handling infrastructure required.

LocALE's automatic service activation and migration capabilities simplify sentient application deployment and enable CPU intensive systems, such as TRIP, to reuse the spare resources available in a network. LocALE offers to applications *location-constrained lifecycle control* over distributed objects. On service activation, LocALE-enabled applications specify the service's initial network location and the constraints under which that service will later be migrated or recovered. As result of this process, LocALE returns to client applications permanent object references that are valid for the entire lifetime of the object, no matter how many times that object is migrated or recovered. A transparent request redirection is performed upon client remote call issue on an object whose location changed. Whereas SIF concentrates on aiding sentient application development, LocALE focuses on making their deployment simpler. A full description of the capabilities and implementation of this middleware software is given at [11].

6. Applications

The TRIP sensor technology has been employed in the development of several sentient applications. In this section, two significant examples of its usability in both off-line and real-time video parsing are given.

6.1. LCE Sentient Library

Conventional tracking technologies attach their active tags only to entities considered more valuable than the tags themselves, e.g. computers or personnel. Moreover, these active tags' size and thickness prevent them from being attachable to small items. TRIP's costless resizable tags and ample range of identifiers make this positioning technology suitable even for the location of small and low-cost items in a physical space. This is the case of stationary (e.g. staplers or hole punchers), or books shared by the personnel in an office. People will often change the location of these items, enforcing other colleagues to search for them through the office. TRIP's versatility, to tag and locate any entity in an environment, has been applied to the problem of finding the location of books shared by researchers in our laboratory.

The *LCE Sentient Library* system augments a conventional library catalogue system with sentient features. Apart from the typical functionality expected in such systems, this one offers contextual information about the books in our lab. Details on the last seen book location and its proximity to other items in the environment are searchable by the user. Figure 8 shows the result of a book search through the LCE Sentient Library web front-end. This application can be viewed online at: http://www-lce.eng.cam.ac.uk/Library.

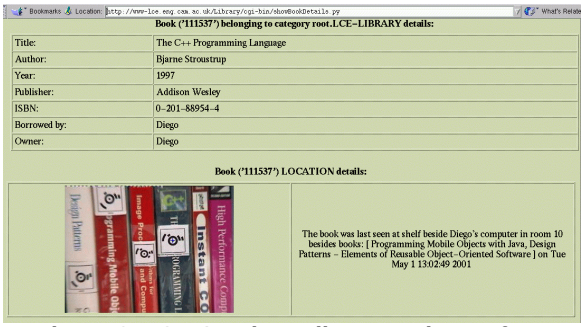


Figure 8. LCE Sentient Library Web Interface

Ideally, the Sentient Library would use cameras spread throughout our laboratory to monitor in real time the movements of TRIP-tagged books. The first prototype, however, uses off-line processing of a video footage with book sightings to automatically update the book catalogue, stored in the TRIP Directory Server.

Periodically, the LCE librarian records with a digital video camera the TRIP-tagged locations and books in our lab. Every location where a book may be located is tagged with a location-type TRIPcode. Similarly, TRIPtags are attached to book spines (see Figure 8). The static data details associated to books and their locations were stored in the TDS using its GUI front-end, as shown Figure 6.

Figure 9 depicts the operation of the Sentient Library system. A Video Frame Server provides access to a serialised book locations' video footage, whereas a TRIParser analyses the frames captured. The TDS is used to update book associated contextual data as result of book sightings. The Library Catalogue Agent is the core component coordinating the catalogue updating process. On initialisation, assisted by LocALE, this agent activates the Video Frame Server and the TRIParser components. A web interface is provided for LCE members to: (1) browse through book categories and the books in a category, (2) perform keyword-based search of books, (3) create new book categories, (4) input new books' details and printoug their associated TRIPtag (5) and modify book details.

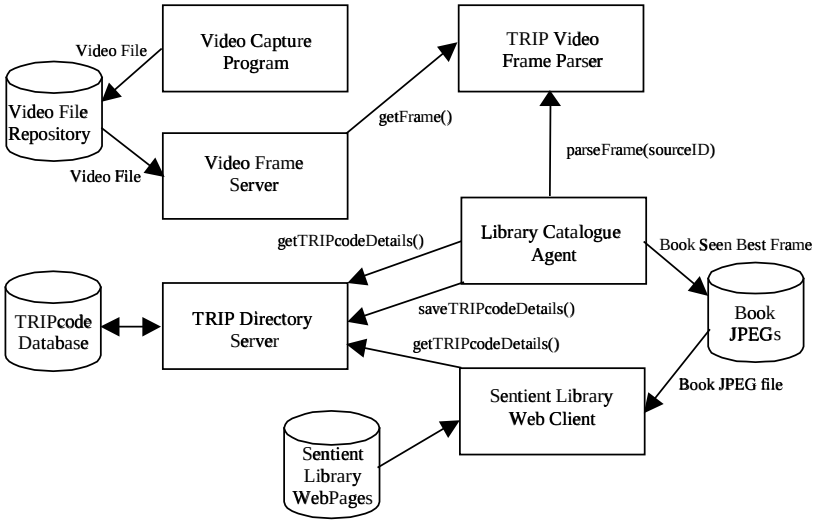


Figure 9. LCE Sentient Library Applicaton Architecture

The off-line video processing nature of this application did not suit well the SIF application construction model. The SIF model is devised for applications requiring real-time notification of contextual information. In this case, the synchronous method interfaces offered by the respective distributed components were employed.

The Sentient Library has been in use for more than 6 months and over time has proved to be a popular tool for the members of our lab. It is conceivable that an enhanced version of the system could be applied to real libraries, where users take books from stands and later, in an attempt to help the librarians, place them back in the wrong location. Librarians could be alerted of books' misplacement while scanning the book stands with a TRIP-enabled camera, rather than requiring them to visually inspect every bookshelf in the library.

6.2 Follow-Me Audio

Sentient spaces attempt to provide to users the software services they require wherever they are. The goal is to allow users to move freely around these spaces without undue degradation of the computing and communications resources available to them. Thus, a common variety of services provided by Sentient Computing are the so-called "follow-me" services. A *Follow-Me Audio* application has been built that demonstrates TRIP's capabilities as a real-time identification and location sensor. The Follow-Me Audio application provides users with music from the nearest speakers wherever they move in our lab. TRIPtag wearers' movements are tracked by analysing the video frames provided by several web-cams installed around our lab. Actions in a distributed jukebox component are issued by showing to the cameras TRIPtags representing jukebox control operations.

Figure 10 shows the different distributed components involved in the implementation of the Follow-Me Audio application. The core component of the system is the Follow-Me Audio Agent, an autonomous active object representing a user. This agent listens to for user movement events or jukebox control actions events provided by the SIF framework, and as a result it migrates the music with the user or it triggers operations on the digital jukebox.

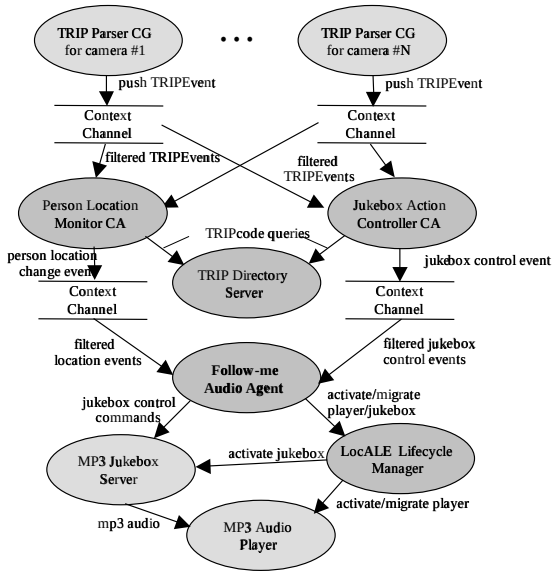


Figure 10. Follow-me Audio Architecture

The Follow-Me Audio Agent operates as follows. Firstly, the agent registers its interest with the *Person Location Monitor CA*'s channel on the movement events of the user. Table 1 shows the registration statement, expressed in the Extended Trader Constraint Language, that the agent for user 'dl231' submits to this CA's channel for the reception of this user's movement events within the LCE premises. Secondly, the agent registers, in a similar way, with the Jukebox Action Controller CA's channel. Upon arrival of the first high-level event coming from the Person Location Monitor CA, the agent activates, through LocALE, the MP3 player and jukebox components. When, subsequently it receives user movement events, the MP3 player is migrated, again using LocALE, to the computer with audio output closest to the user. When the user moves out of the premises the agent destroys both the MP3 player and jukebox server. On arrival of events from the Jukebox Controller CA, the agent issues the suitable control command to the jukebox server.

```
struct MovementEvent {
   string personID;
   locDesc from; // e.g. 'building.floor.room.region'
   locDesc to;  // e.g. CUED.4.LCE-Room1.camera1
};
```

```
( ( ($domain_name == 'PersonLocationMonitor') and
    ($type_name == 'Movement') and
    ($personID == 'dl231') and
    ('LCE' ~ $from)  // is 'LCE' a sub-string of $.from?
  )
)
```

Table 1. Filtering expression on movement event

Both Person Location Monitor and Jukebox Action Controller CAs undertake a similar function. They register with the TRIParser CGs' context channels to receive the TRIPevents corresponding to their TRIPcode categories of interest. Then, they interpret the events received, with the assistance of the TRIP Directory Server, and produce the high-level events that can directly drive the personal agent operation.

As the user wanders around the location-aware environment, the music follows him. The state of the system and time index into the current song persists as the components migrate. This application leverages TRIP tracking capabilities, the SIF application construction model and LocALE's migration support.

7. Related Work

SONY's CyberCode [20] is a visual tagging system based on a 2-D square barcode that alike TRIP can be used to determine the 3-D position and identifier of tagged objects. The ARToolKit [9] system also chooses a square marker for a similar purpose, however, its identifier encoding capabilities are more limited than in the case of TRIP or CyberCode. Both CyberCode and ARToolKit technologies provide a high degree of location accuracy and work in real-time. They have been mainly applied to the domain of Augmented Reality, in order to correctly register computer-synthesised information on views of the real work. However, these square marker technologies are not as easily identifiable as TRIP circular targets in cluttered environments and therefore are less suitable for the object location and tracking domain we target. Moreover, their marker geometric features require higher image resolution for accurate recognition and location extraction than TRIP.

BBC's free-d [23] location system measures the precise position and orientation of studio cameras, by using an auxiliary camera mounted on the back of a conventional moving camera pointing to circular markers, similar to TRIPcodes, fixed on the ceiling of a TV recording studio. A hardware implementation of its algorithms is needed to achieve real-time video processing. The system is used for virtual reality TV production, being, in contrast to TRIP, expensive and cumbersome to deploy.

GeorgiaTech's Context Architecture project [4] attempts to make sentient application development as simple as GUI development. For that it introduces Context Widgets that separate context sensing from context-use. SIF abstractions present certain resemblance to the ones proposed by the Context Architecture. However, SIF focuses more on efficient context information dissemination. Microsoft's EasyLiving [3] project creates reactive context-aware living spaces without the user having to wear any location tag or computing device. Their approach to track people based on colour histograms, without requiring the user to wear any marker, has much heavier computational demands and produces less reliable results than TRIP. AT&T's Sentient Computing project [1] aims to replace human computer direct interaction (through mouse or keyboard) by enabling users to instead interact with their surrounding space based on the precise location and orientation provided by the Active Bat Location System. This concept has been illustrated with several sophisticated sentient applications [7].

8. Further Work and Conclusion

Thus far, only the entity presence identification capability of TRIP has been exploited. Future work will address the creation of sentient applications that make use of the 3D location and orientation information also provided by TRIP.

Presently, we are working on the idea of making sentient application creation feasible even for computer illiterate users. We believe it is key the involvement of end-users in the definition of rules that delimit their expectations from the reactive environment. Only in this way, sentient systems will respond to users with the right actions at the

right time, satisfying end-user individual needs and making interactions more satisfactory and undisruptive. Our ongoing research proposes the association of every user in a sentient environment with an Event-Condition-Action Agent, ECAgent in short. ECAgents embody a set of *situation-action* rules that govern the interactions of the user with a Sentient Space. Some progress has already been done in the automatic code generation of these ECAgents given a set of IF-THEN rules specified by the user. A key feature of our approach is that we map the user input rules into a production systems' programming language's rules, CLIPS [15] specifically. This rules are passed to a CLIPS inference engine embedded in each agent that undertakes the rule-based reasoning. The most challenging part of this work appears to be the design of a GUI that enables the creation of generic event condition action rules.

TRIP is a novel cost-effective and easily deployable location sensor technology. This sensor's off-the-shelf hardware requirements, i.e. inexpensive CCD cameras and CPU processing, makes the creation of location-aware reactive environments, even in the home, an affordable proposition. All that is required to augment a standard PC with visual awareness is a web-cam and TRIP's software. SIF is an application construction model that eases the development of sensor-driven systems and efficiently manipulates and disseminates context information. LocALE is an object lifecycle and location control middleware that streamlines user-bound service activation, migration and deactivation. In addition, it permits application developers to reuse the spare networked computing resources in a LAN. LocALE complements TRIP's goal of minimising the investment cost and deployment complexity in the construction of sentient spaces. A couple of TRIP-enabled applications have been illustrated to validate our contributions.

Acknowledgements

The author is very grateful to the Department of Education of the Basque Government for the funding of his PhD studies and to AT&T Laboratories Cambridge for sponsoring the TRIP project. Thanks are also due to R.S. Sohan for his help in the deployment of the TRIP system throughout the LCE laboratory and to K. Sanmugalingam for proof-reading this article.

References

[1] AT&T Laboratories Cambridge, "Sentient Computing Home Page", 2001, http://www.uk.research.att.com/spirit/

[2] Brown P.J., Bovey J.D. and Chen X. "Context-aware Applications: from the Laboratory to the Marketplace", IEEE Personal Communications, 4(5), pp. 58-64, 1997

[3] Brumitt B., Meyers B., Krumm J., Kern A., Shafer S. "EasyLiving: Technologies for Intelligent Environments", Handheld and Ubiquitous Computing, September 2000

[4] Dey A.K., Salber D., Futakawa M. and Abowd G. "An architecture to support context-aware applications", 12th Annual ACM Symposium on User Interface Software and Technology, 1999

[5] Forsyth D., Mundy J.L., Zisserman A., Coelho C., Heller A. and Rothwell C. "Invariant Descriptors for 3-D Object Recognition and Pose", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, No. 10, pp. 971-991, October 1991

[6] Harter A. and Hopper A. "A Distributed Location System for the Active Office", IEEE Network, Vol. 8, No. 1, January 1994

[7] Harter A., Hopper A, Steggles P., Ward A. and Webster P. "The Anatomy of a Context-Aware Application", Proceedings of MOBICOM'99, Seattle, August 1999

[8] Hopper. A. "The Clifford Paterson Lecture, 1999, Sentient Computing", Philosophical Transactions Royal Society London. A (2000) 358, 2349-2358.

[9] Kato H. and Billinghurst M.. "Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System", Proceedings of the 2nd Internation Workshop on Augmented Reality, pp. 85-94, October 1999

[10] Korkea-aho M. "Context-Aware Applications Survey", Internetworking Seminar (Tik-110.551), Helsinki University of Technology, 2000.

[11] López de Ipiña D. and Lo S. "LocALE: a Location-Aware Lifecycle Environment for Ubiquitous Computing", Proceedings of the 15th International Conference on Information Networking (ICOIN-15), February 2001

[12] López de Ipiña D. "TRIP: a Low-Cost Vision-Based Location System for Ubiquitous Computing", submitted to the 2001 Workshop on Perceptive User Interfaces, November 2001.

[13] López de Ipiña D., "TRIP: A Distributed vision-based Sensor System", Laboratory for Communication Engineering, Cambridge, Technical Report, September 1999

[14] Moran T.P., Saund E., Van Melle W., Gujar A.U., Fishkin K.P. and B.L. Harrison. "Design and technology for Collaborage: collaborative collages of information on physical walls", Proceedings of the 12th annual ACM symposium on User interface software and technology, November 1999

[15] NASA, "CLIPS: A Tool for Building Expert Systems", http://www.ghg.net/clips/CLIPS.html, August 1999

[16] Object Management Group, "Trading Object Service Specification", May 2000

[17] Object Management Group, "Naming Service Specification", February 2001

[18] Object Management Group, "Notification Service", November 1998

[19] Object Management Group, "The Common Object Request Broker Architecture: Architecture and Specification", October 1999

[20] Rekimoto J. and Ayatsuka Y. "CyberCode: Designing Augmented Reality Environments with Visual Tags", Proceedings of DARE 2000, 2000

[21] Salber D., Dey A. K. and Abowd G. D. “The Context Toolkit: Aiding the Development of Context-Enabled Applications”, Proceedings of CHI'99, May 1999

[22] Stafford-Fraser J.Q and Robinson P. “BrightBoard: A Video-Augmented Environment”, Proceedings of Conference on Human Factors in Computing Systems (CHI96), April 1996

[23] Thomas G. A., Jin J., Niblett T., Urquhart C. “A versatile camera position measurement system for virtual reality in TV production”, Proceedings of IBC’97, pp. 284-289, September 1997

[24] Tsai R. Y. "A versatile Camera Calibration Technique for High-Accuracy 3-D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses", IEEE Journal of Robotics and Automation, vol. RA-3, no. 4, pp. 323-344, August 1987. Source code at: http://www.cs.cmu.edu/afs/cs.cmu.edu/user/rgw/www/TsaiCode.html

[25] Want R., Hopper A., Falcão A. and Gibbons J. “The Active Badge Location System”, ACM Transactions on Information Systems, Vol. 10, No. 1. 91-102, January 1992

[26] Ward A., Jones A. and Hopper A. “A New Location Technique for the Active Office”, IEEE Personal Communications, October 1997, pp. 42-47

[27] Weiser M. “The Computer for the 21st Century. Scientific American, 265(3):94-104, September 1992