Sentient Computing for Everyone

Diego López de Ipiña (1) and Sai-Lai Lo (2)
(1) Laboratory for Communications Engineering, Department of Engineering, University of Cambridge, Cambridge, UK
(2) AT&T Laboratories Cambridge, 24a Trumpington Street, Cambridge, UK

Key words: Context-Aware Computing, Location-Aware Computing, CORBA, Mobility

Abstract: Sentient Computing gives perception to computing systems so that they can detect, interpret and respond to changing aspects of users' context. The location attribute of a user's context is of special interest because it makes human-computer interactions more natural. This explains the appearance in the last years of several sophisticated indoor location technologies to track user whereabouts. However, these positioning systems are costly and difficult to deploy, configure and operate, having prevented a wider adoption of the Sentient Computing paradigm. This paper describes a novel vision-based software location system, known as TRIP, whose low-cost, off-the-shelf hardware requirements and easy deployment features overcome other systems' limitations. Nevertheless, in order to foster the deployment of sentient spaces, bringing services to users wherever they are or move to, a location system must also be accompanied by middleware to facilitate user-bound software services activation, movement and destruction. LocALE, a CORBA-based software that addresses heterogeneous object lifecycle and location control in a network, is our solution to this issue. Some distributed applications combining TRIP and LocALE's capabilities are presented to demonstrate that Sentient Computing can be made readily available for everyone and everywhere.

1. INTRODUCTION

Ubiquitous Computing [16] envisions physical spaces, such as offices, meeting rooms or homes, that are augmented with computing devices integrated into the environment. It aims to make services provided by these

devices as commonly available as electricity. With Ubiquitous Computing, personal computers lose user focus of attention since the computational environment is spread across the physical space. The user perceives just the functionality, not the invisible devices providing it.

Sentient Computing [5] is our approach to make Ubiquitous Computing a reality. It stems from the basic idea that to make computerised services pervasive in our life, the devices providing such services must be given *perception*, i.e. the capability to *see* or *hear* what entities are around them, what those entities are doing, where they are and when something is happening. Only then it is possible for the underlying computing systems to undertake suitable actions matching end user expectations. For example, when a user enters in his office, depending on the time and day of the week, the sentient environment could automatically initiate the playback of a specific kind of music. If a colleague later on comes into the room to discuss about some project ideas, the music volume level could be automatically adjusted to facilitate communication. Sentient Computing creates perceptive, richly computerised environments that consume data from sensors spread throughout a physical space and act upon such impulses, e.g. automatic service activation, personalisation or adaptation on the basis of previously specified users' preferences.

Location-Aware Computing [9], whose behaviour is determined by the position of objects in the environment, represents an interesting subset of the Sentient Computing paradigm since location is often an essential attribute of context. This has motivated the appearance of several indoor location systems providing different entity location granularity, i.e. ranging from room-scale resolution such as the infrared-based Active Badge [15], to more accurate 3D coordinate resolution offered by systems such as the ultrasonic-based Active Bat [4] or the radio-based 3D-iD [17]. All these systems require *locatable* people and objects to be attached an electronic tag that transmits a unique identifier via either an infrared, ultrasound or radio interface to a network of sensors in 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. However, the existing systems present some inconveniences. The tags they use need battery power and are expensive, and the infrastructure required, a network of sensors, is complex to install and maintain and also expensive. These factors have constrained the deployment of such tracking systems to organisations with enough money and qualified personnel to install and keep the system running. We have devised an alternative sensor technology, named TRIP that aims a better trade-off between the price and flexibility of the technology and the accuracy of the location data provided. TRIP is a vision-based sensor technology that recognises and locates passive printed 2D circular barcode

tags, attached to objects, when viewed from inexpensive low-resolution CCD cameras.

Nevertheless, in order to promote the adoption of the Sentient Computing paradigm in our living spaces, it is not only sufficient with making computing systems aware of user presence, movement or precise location, but it is also required that these systems are provided with the capability of automatically activating services on behalf of the user when appropriate contextual conditions are met. Furthermore, it is desirable to enable user-bound services to follow the user as he moves through the physical space, and to deactivate such services when the user leaves the sentient premises. In conclusion, what is needed is a middleware infrastructure that gives sentient applications control over object's lifecycle and location in a network. The LocALE framework is our middleware solution to this need. Sections 2 and 3 explore the TRIP technology and the distributed systems infrastructure built on top of it. LocALE is overviewed in section 4. Section 5 describes some applications combining TRIP and LocALE. Section 6 summarizes some related research. Finally, section 7 draws some conclusions.

2. TRIP: A DOWNLOADABLE LOCATION SENSOR

TRIP (Target Recognition using Image Processing) [8] is a novel vision-based sensor system that uses a combination of visual markers (2-D *ringcodes*) and conventional video cameras to identify tagged objects in the field of view. Relatively low CPU demanding image processing and computer vision algorithms are applied to video frames to obtain the identifier (*TRIPcode*) and pose (location and orientation) of the targets with regard to the viewing camera.

TRIP constitutes a very cheap and versatile sensor technology. Its 2-D ringcodes are printable, and can therefore be attached even to low-cost items, such as books or office stationary (e.g. stapler). A TRIPcode (see Figure 1), read in counter-clockwise fashion from its synchronisation sector, represents a ternary number in the range $1\text{-}3^{13}$ (1,594,323). Only off-the-shelf hardware is required: low-resolution CCD cameras and CPU processing power. The TRIP software is planned to be made publicly downloadable in due time so that users with a web-cam attached to their PCs may easily install and run this software sensor and hence provide visual awareness to their computers.

A Target Recognition Algorithm is applied to the video data provided by a camera that executes a set of image processing stages in order to determine the identifier and location in the image (coordinates x,y) where TRIPcodes are spotted. [8] fully describes the mathematics involved in the process. On the other hand, a Pose Extraction Algorithm is used that given a single view

of a target, determines 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. The method applied is based on the POSE_FROM_CIRCLE method given at [3].

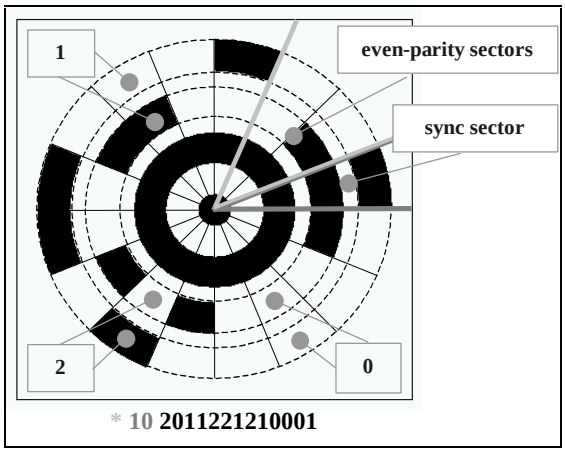

Figure 1. TRIPcode representing number 1,160,407

The C++ implementation of the 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 is about 12Hz. TRIPtags are recognised as long as a target's plane orientation angles are less than 70º and its image occupies at least 20x20 pixels. The 3D location of the centre of a target with regard to a viewing camera is obtained with less than 5% error. An even error parity check (see Figure 1) is used to prevent the identification of false positives.

3. TRIP: A DISTRIBUTED SENSOR SYSTEM

An event-based distributed architecture has been devised around TRIP in order to manipulate and distribute the sensor data provided by this technology. The functionality of the TRIP system, i.e. its target recognition and pose extraction, has been encapsulated within a CORBA [11] component named *TRIParser*. This component offers a UNIX `pipe`-like interface that enables applications to connect distributed *Frame Grabber* components, obtaining images from cameras spread throughout the environment, to TRIParsers. A TRIParser may consume images supplied by one or more Frame Grabbers. TRIP processing results are, by default, asynchronously communicated in event-form to a CORBA Notification

Channel [10] associated with each TRIParser. This interleaved event communication component decouples analysers' frame processing and result reporting duties.

A TRIParser pushes *TRIPevents* (see Figure 2) to its associated Notification Channel. Parties interested in a given TRIParser's location data subscribe to its associated channel and convey a set of constraints that specify the events they are interested in receiving. A Notification Channel undertakes consumer registration, event filtering and communication on behalf of its representing TRIParser. The filter constraint language supported is the constraint language defined by the OMG Trading Service. Hierarchical interconnections of TRIP Parsers' Notification Channels can be created in order to ensure the efficient and scalable dissemination of TRIP generated sensorial data. The TRIParser also provides a synchronous invocation interface (`processFrame`) that analyses a submitted frame and returns the location data inferred from it. Hence, applications can interact with a TRIParser in either a synchronous or asynchronous form. However, for efficiency purposes, applications should establish direct communication pipes between Frame Grabbers and TRIParsers, and register as event consumers of the parsers' Notification Channels.

```
struct TRIPevent {
  string TRIPcode; // code ternary representation
  string cameraID; // capturing camera identifier
  paramsEllipse params; // bull's-eye's outer ellipse parameters (x,y,a,b, θ)
  targetPosition pose;  // translation vector from camera origin to target centre
  targetOrientation angels; // (α, β, γ)
};
```

Figure 2. TRIPevent contents

3.1 The TRIP Directory Server

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. The TDS offers CORBA interfaces for the creation, modification, deletion and retrieval of both TRIPcodes and their categories.

3.2 The Sentient Information Framework

The Sentient Information Framework (SIF) defines an application construction model to streamline sentient applications development. SIF

isolates context capture and abstraction from application semantics and, at the same time, it provides efficient mechanisms for context communication. Its main function is to massage context information into the formats demanded by applications. SIF is not constrained to TRIP, being sensor-technology independent. In this framework, components are categorised in 3 types (see Figure 3): (1) Context Generators (CGs), (2) Context Channels (CCs) and (3) Context Abstractors (CAs). Context Generators encapsulate sensors, such as TRIP, and the software that extracts information from them, and are sources of sensorial events. Context Channels, in our case implemented as CORBA Notification Channels, receive events and, after consumer specified filtering, pass corresponding events to registered parties. Context Abstractors achieve the separation of concerns between context sensing and application semantics. They consume the 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 can directly drive applications. Sometimes they also *correlate* other CAs' or CGs' outcomes to generate the required sentient data. For a more detailed description of SIF refer to [6].

4. LOCALE: MIDDLEWARE FOR OBJECT LIFECYCLE AND LOCATION CONTROL

In order to support our thesis that sentient computing can be made available for everyone everywhere, we have, so far, illustrated a new cost-effective and easily deployable location sensor technology, TRIP, and a sentient application construction model, SIF, for the efficient manipulation and dissemination of sensorial data. One question still to be answered, however, is how, once a sentient system receives a high level event (e.g. Diego enters in his office), can the system effectively trigger the required 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 streamline 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 object factory processes 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 user location, e.g. the activation of an MP3 player must take place in one of the PCs of a room where a user presence is detected. Therefore, it is desirable to control both the lifecycle and location of user-bound services in the network.

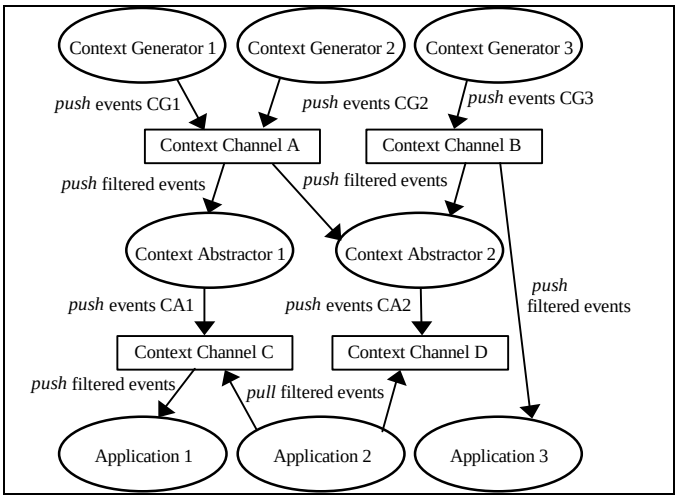

Figure 3. The SIF Architecture

The LocALE (Location-Aware Lifecycle Environment) framework [7] defines a simple mechanism for managing the lifecycle and location of distributed CORBA objects residing in a network. In addition, it provisions with load-balancing and fault-tolerance features to the objects whose lifecycle manages. 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 and capabilities, can intelligently direct components' locations and lifecycles. LocALE offers an object-lifecycle handling infrastructure, simplifying sentient application deployment and enabling CPU intensive systems, such as TRIP, to reuse the spare resources available in a network.

4.1 LocALE Architecture

The LocALE middleware presents a 3½-tier architecture (Figure 4) composed of client applications and the following 3 types of components:

– The *Lifecycle Manager* (LCManager) provides object-type independent lifecycle control interfaces and mediates the lifecycle operations over any CORBA object residing in a location domain. A location domain is a group of machines on a LAN located within a given physical area, such as a building, a floor or a room. Every object creation, movement or deletion request is routed through this component. This permits the LCManager to cache the current location of every object in a domain and thus act as a forwarding agent that redirects client requests after object references held by clients are broken due to either object movement or

failure. In the latter case, the LCManager first tries to recover the failed object, and, in case of success, returns the new object reference.

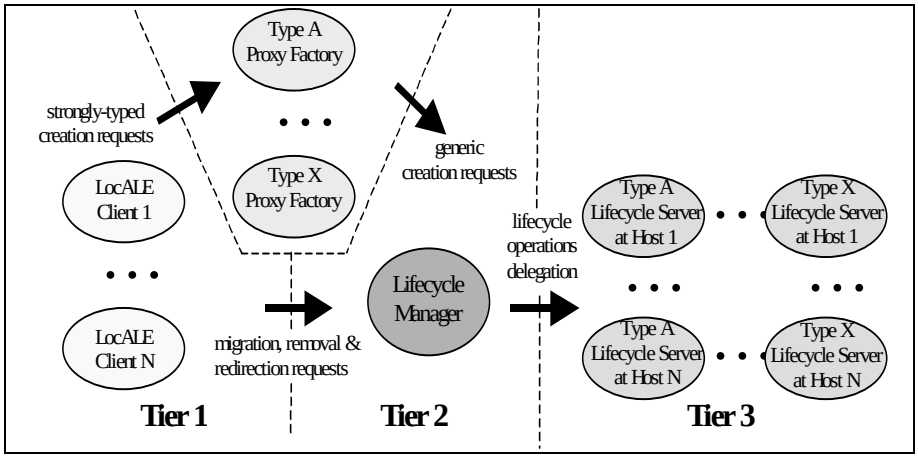

Figure 4. LocALE 3½-tier architecture

– *Lifecycle Server(s)* (LCServer) are containers of LocALE-enabled objects subject to lifecycle control. Lifecycle operations invoked on an LCManager are delegated to suitable LCServers. They subsume standard strongly typed factories' functionality by providing a type specific creation method with hard-coded types. Furthermore, they also assist their local objects on their migration to other LCServers. They can be started either manually or by the local LCManager after a creation or migration request arrives at a location where the required LCServer type does not exist. In either case, they register with the manager by passing their physical location (host), their CORBA object reference (or IOR) and data specific to the type of objects they handle.
– *Type-specific Proxy Factories* are placed in between clients and LCManagers. They prevent client applications from dealing with the generic object creation interface offered by an LCManager. Using specific-purpose interfaces makes client code far shorter and simpler to understand. With the generic version if a mismatch in the type of a constructor argument occurred, the client would only receive an error at run-time rather than at compile time. Type-specific Proxy Factories offer type specific object-creation methods with the same argument types and semantics as the LCServers they represent. They map type specific requests to the generic format demanded by LCManagers. They are on demand activated by the local LCManager.

LocALE's architecture guarantees clients' compile-time type safety with respect to the lifecycle control of their required services. Clients find, through the LCManager, object references to type-specific proxy factories

and issue through them object creation requests. Object migration and deletion requests are directly invoked on the LCManager because they are object type-independent. The LCManager then delegates incoming lifecycle operations to the appropriate LCServers.

4.2 Location-constrained Lifecycle Control

One of the main advantages of CORBA [11] is that it offers location transparency to applications, i.e. it makes method invocation as simple on remote objects as on local objects. LocALE leverages CORBA's location transparency, but it asserts that being able to control *where* services used by clients are located, and also to set the constraints under which those services can later be re-located may bring important benefits. For example, load-balancing systems may wish to initiate new service instances on the hosts of a LAN with the lowest processing load. Follow-me sentient applications may want to move objects tied to a user's physical location to the nearest host with the required capabilities. LocALE provides CORBA objects' *location-control* to applications. For that it changes distributed object construction semantics compared with conventional factory objects. The following two attributes are appended to the object creation interfaces offered by proxy factories:

– *Location* attribute (`LocSpec`): specifies *"where"* a service should be instantiated (or moved to). The formats available for location specification are: (1) `hostDN(hostName)`, (2) `roomID(room)` and (3) `hostGroup(listHostName)`. `hostDN` and `roomID` are used when the creation of an object is wished in a specific host or room's host. By setting `hostName` or `room` attribute to `"ANY"` in either of these `LocSpecs`, the programmer can instantiate location-independent services. The `hostGroup` format is useful to specify an arbitrary set of hosts where an object may be created, e.g. hosts in a room with audio capabilities.
– *LifeCycle contraints* (`LCconstraints`) attribute: determines whether a creating object should be considered as `RECOVERABLE` and/or `MOVABLE` within the location scope in which it was created, i.e. within a host, room or a host group. A recoverable object is either a stateless or a persistent object whose state can be restored after failure.

The capability of receiving location constraints on object creation converts the LCManager into a minimal Trader server. The LCManager matches client object creation specifications against the registered object Lifecycle Server types and locations, and delegates the operation to a suitable LCServer.

4.3 LocALE Implementation

The LCManager has been implemented in C++ using the CORBA 2.3 C++ ORB omniORB [1]. Its implementation makes extensive use of some advanced features provided by the CORBA 2.3's Portable Object Adapter (POA). Among these, it is essential the POA defined standard mechanism to generate LOCATION_FORWARD reply messages. These messages are sent by LCManagers to client ORBs to indicate the new location of an object where previously issued requests have to be redirected. A detailed description on the implementation details of the LocALE middleware is offered in [7].

5. TRIP-ENABLED SENTIENT APPLICATIONS

This section describes three sentient applications[1] that combine TRIP sensing with LocALE's object lifecycle and location management. They demonstrate that our two contributions streamline sentient systems' deployment and make it a cost-effective process.

5.1 The LCE Sentient Library

This system augments a conventional library catalogue system with sentient features. Apart from the typical functionality expected in a book cataloguing system, the LCE Sentient Library offers contextual information about books in our lab. Details on the last seen book location and its proximity to other fixed items in the environment are offered. This application is an illustration of TRIP versatility for tracking any tag-able entity in the environment.

Every location where a book may be located in our lab has been tagged with a location-type TRIPcode. Similarly, book-category TRIPtags have been attached to book spines. Periodically, the LCE librarian wanders around our lab with a digital camera recording TRIP tagged locations and books. The system automatically updates the book database by the off-line processing of a previously recorded book sightings video. This process involves the co-operation of a Video Frame Grabber, providing access to the serialised video's frames, a TRIParser, analysing those frames, and the TRIP Directory Server, where book details and contextual data is recorded. The Video Frame Grabber and TRIParser components are automatically activated by LocALE.

[1] The first two applications are available at: http://www-lce.eng.cam.ac.uk/~dl231/#Demos.

A web interface provides mechanisms for LCE members to (1) browse through all the book categories, (2) books in a category, (3) given book details and (4) perform keyword-based search of books. Figure 5 shows the result of a book search in the LCE Sentient Library.

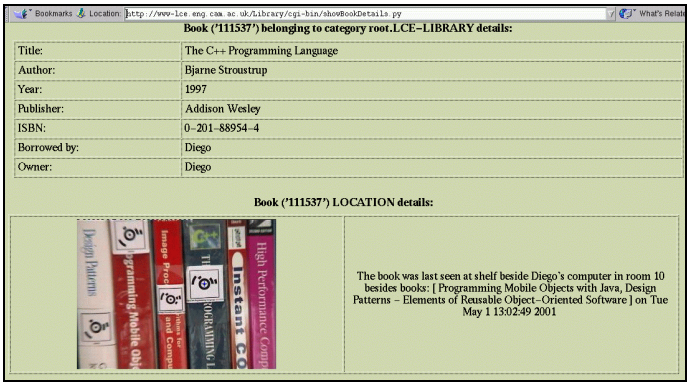

Figure 5. LCE Sentient Library snapshot

5.2 Active TRIPboard

This application augments a conventional whiteboard with interactive commands issued by placing TRIPcodes in the field of view of a camera observing the whiteboard. Some example actions that can be triggered are: (1) capture whiteboard snapshot and email a web link to it to people currently present in the room, (2) printout a copy of the whiteboard snapshot for each person in the room. The application components: a Frame Grabber and a TRIParser are activated via LocALE either upon person presence in our meeting room or through a web interface (see Figure 6). The Active Badge indoor location system deployed in our lab is used for person presence detection. In future work our meeting room will be populated with sufficient cameras to cover all the field of view of the room, and so permit TRIPtag wearer presence detection. Through LocALE, the TRIParser is activated in a load-balanced way by random selection of one of the hosts provided in a `hostGroup`. If the TRIParser fails, the application recovers transparently by parser recreation through LocALE's fault-tolerance support.

5.4 Follow-me Audio

This application provides mobile users with music from the nearest speakers wherever they move in our lab. The source of music is an MP3

jukebox server whose operation is controlled by showing jukebox-type TRIPtags to web-cams attached to some of our lab's PCs. A personal software agent, associated with each lab person, listens for that person's movement events generated by a Person Location Monitor Context Abstractor. This component is registered with the Context Channels of the TRIParsers processing PC web-cams' frames. From raw TRIP sighting events, it generates person presence and movement events. The personal agent, acting as a component migration controller, requests LocALE to migrate the audio player and MP3 decoder components of the application to the host nearest to the user supplying the appropriate sound facilities. 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, SIF context modelling features and LocALE's migration support.

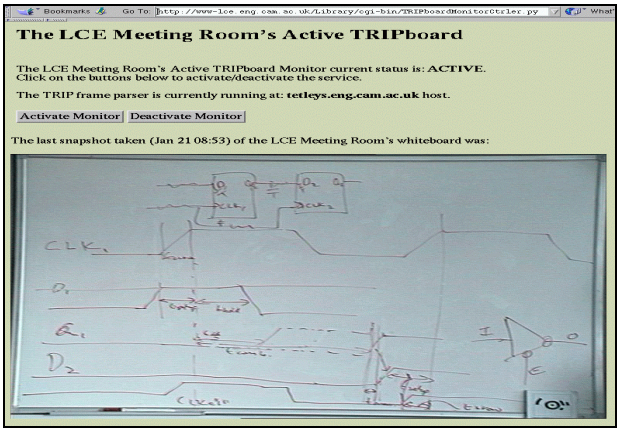

Figure 6. Active TRIPboard snapshot

6. RELATED WORK

SONY's CyberCode [12] is a visual tagging system based on a 2D square barcode that can be used to determine the 3D position and identifier of tagged objects. Several augmented reality applications have been produced based on this system, e.g. the visitor view of CyberCode tagged items in a museum is augmented with synthesised information. The geometric features of CyberCodes require higher image resolution for their accurate recognition and location than TRIP. BBC's free-d [14] 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, alike to TRIPcodes, placed on the ceiling of a TV recording studio. A hardware implementation of its algorithms was necessary to achieve real time processing. The system is used for virtual reality TV production, being expensive and cumbersome to deploy.

GeorgiaTech's Context Architecture project [2] attempts to make sentient application development as simple a GUI development. For that it introduces Context Widgets that separate context sensing from context-use. It shares some similarities to SIF but doesn't tackle efficient context information dissemination. Microsoft's EasyLiving [13] project tries to create 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 [5] 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 [4]. This concept has been illustrated with several sophisticated sentient applications.

7. CONCLUSION

TRIP, a novel cost-effective and easily deployable location sensor technology, has been introduced. This sensor's off-the-shelf hardware requirements, i.e. inexpensive CCD cameras and CPU processing, makes affordable the creation of location-aware reactive environments, even in the home. All that is required to augment a standard PC with visual awareness is a web-cam and TRIP's downloadable software. SIF, an application construction model to ease the development of distributed sensor-driven systems has also been described. LocALE, an object lifecycle and location control middleware that streamlines user-bound service activation, migration and removal and, at the same time, permits application developers to reuse the spare networked computing resources in a LAN, has been presented. LocALE complements TRIP's goal of minimising the investment cost and deployment complexity involved in the construction of sentient environments. LocALE is a very useful tool for sentient applications that have to trigger user-related services in response to captured sensor data. Several TRIP-enabled applications, following the SIF model and leveraging from LocALE infrastructure have been developed for the sentient office scenario, thus validating our contributions.

ACKNOWLEDGEMENTS

The authors are very grateful to the Basque Government's Department of Education and AT&T Laboratories Cambridge for their sponsorship.

REFERENCES

[1] AT&T Laboratories Cambridge, "omniORB C++ CORBA ORB", http://www.uk.research.att.com/omniORB/omniORB.html

[2] Dey A.K., Salber D., Futakawa M. and Abowd G. "An architecture to support context-aware applications", 12th ACM's UIST, 1999

[3] 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

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

[5] Hopper. A. "Sentient Computing", The Royal Society Clifford Paterson Lecture, AT&T Laboratories Cambridge, Technical Report 1999.12, 1999

[6] López de Ipiña, D. "Building Components for a Distributed Sentient Framework with Python and CORBA ", Proceedings of the 8th International Python Conference, January 24 - 27, 2000

[7] López de Ipiña, D. and Lo S. "LocALE: a Location-Aware Lifecycle Environment for Ubiquitous Computing", Proceedings of ICOIN-15, February 2001

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

[9] Nelson G. "Context-Aware and Location Systems". PhD Thesis. Cambridge University Computer Lab, UK, January 1998

[10] Object Management Group. "Notification Service Specification", June 2000

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

[12] Rekimoto J. and Ayatsuka Y. "CyberCode: Designing Augmented Reality Environments with Visual Tags", Designing Augmented Reality Environments (DARE 2000), 2000

[13] Shafer S, et al. "The new EasyLiving Project at Microsoft Research, Microsoft, 1999

[14] 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, September 1997

[15] 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

[16] Weiser M. "The Computer for the 21st Century". Scientific American, September 1992

[17] Werb J. and Lanzl C. "Designing a positioning system for finding things and people indoors", IEEE Spectrum, pp.71-78, September 1998