TRIP: A Distributed vision-based Sensor System

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

Abstract

This report describes the development of a novel sensor technology named TRIP (*Target Recognition using Image Processing*) that uses the combination of visual markers (2-D circular bar code tags) and video cameras to automatically identify tagged real word objects in the field of view. A CORBA event-based distributed component architecture employed to manage and distribute to applications the sensorial data provided by TRIP and an example application that benefits from it are also overviewed. Finally, extensions to this sensor technology and potential applications are proposed and a schedule is given of how the author will proceed to complete this research in his remaining two years of PhD.

Contents

1. Introduction 1
  1.1. Research Motivation 2
  1.2. Research Statement 2
  1.3. Report Outline 2
2. Related Work 3
  2.1. Location Technologies Overview 3
    2.1.1. Infrared-based Technologies 3
      2.1.1.1. Active Badge 3
      2.1.1.2. PARCTAB 3
      2.1.1.3. Smart Badge 4
      2.1.1.4. Locust Swarm 4
    2.1.2. Ultrasound-based Technologies 5
      2.1.2.1. Active Bat 5
    2.1.3. Radio-based Technologies 5
      2.1.3.1. RFID tags 5
      2.1.3.2. GPS 6
    2.1.4. Vision-based Technologies 6
      2.1.4.1. Pfinder 6
      2.1.4.2. SONY Navicam 6
  2.2. Management of Context Information 7
    2.2.1. The Active Badge Distributed Location Service 7
    2.2.2. The Active Map Location Service 7
    2.2.3. The Situated Computing Service (SitComp) 8
    2.2.4. SPIRIT (Spatially Indexed Resource Identification and Tracking) 8
    2.2.5. The Context Architecture 9
    2.2.6. The stick-e note infrastructure 9
  2.3. Application Areas 10
3. TRIP: a vision-based sensor technology 13
  3.1. A 2-D barcode as a Location Device 14
    3.1.1. TRIP Target Design Issues 14
  3.2. Target Recognition Process 15
    3.2.1. Image Acquisition (Stage 0) 15
    3.2.2. Image Contrast Enhancement (Stage 1) 15
    3.2.3. Edge Enhancement (Stage 2) 17
    3.2.4. Edge Localisation (Stage 3) 17
    3.2.5. Edge Following and Filtering (Stage 4) 17
    3.2.6. Ellipse Detection (Stage 5) 18
    3.2.7. Concentric Test (Stage 6) 18
    3.2.8. Code Recognition (Stage 7) 18
4. TRIP Sensor System Evaluation 21
  4.1. TRIP Accuracy and Resolution 21
  4.2. TRIP Performance 22
  4.3. TRIP Limitations 22
5. A Distributed Sentient Information Service for TRIP 23
  5.1. Overview of CORBA technology 24
  5.2. The Sentient Information Service (SIS) Architecture 25
    5.2.1. The TRIP Monitoring Service Context Generator 26

5.2.2. Sentient Information Service TRIP-aware Context Abstractors 27
5.2.3. The Jukebox controller TRIP-aware Application 28
5.2.4. The TRIP Directory Service Component 28
5.2.5. A GUI front-end for the TRIP Directory Server 30
6. The Sentient Information Service Architecture Evaluation 32
7. Future work 34
7.1. TRIP Sensor Technology Enhancements 34
7.1.1. Improving TRIP Sensor Recognition Rate 34
7.1.2. Improving TRIP Sensor Location Resolution 34
7.1.3. Target Redesign Considerations 35
7.2. Sentient Information Service Architecture Improvements 35
7.3. Applications in mind 36
7.4. Schedule of Work to be completed 38
8. Conclusion 39
Acknowledgements 39
Appendix A. TRIP Directory Service Implementation 40
A.1. TRIP Directory Server Persistent Dictionaries 40
A.2. TRIP Directory Server Functionality 41
References 42

1. Introduction

Sentient Computing, more commonly known as *context-aware computing* [Schilit94a], concerns the ability of computing devices to *detect*, *interpret* and *respond* to aspects of the user's local environment. Its goal is to enhance computer systems with a sense of the real world and make them know as much as the user about the aspects of the environment relevant to their application. To achieve such purpose it employs sensors distributed throughout the environment to maintain a detailed model of the real world and make it available to applications. Applications can then respond to environmental changes and autonomously change their functionality, without explicit user intervention, based on observations of *who* or *what* is around them, what they are doing, *where* they are and *when* is happening

Research on Sentient Computing is driven by the emergence of low-cost and thus potentially widely available technologies that can provide inputs about the environment. Cameras, microphones or Location Systems such as the Active Badge [Want92] and Global Positioning System (GPS) [Dana98] are possible sources of sentient data. By networking large numbers of such physical sensors and in addition, acquiring through telemetry software information regarding the current state of computers, storage devices and networks, is possible to conceive applications and devices that are highly reactive to the changing state of the physical world. Context-aware applications and devices might personalise themselves to their current user, alter their functionality based on where they were being used, or take advantage of nearby computing and communications resources.

Location-aware systems [Nelson98], whose behaviour is determined by the position of objects in the environment, represent an interesting subset of the sentient computing paradigm since location is often a sine qua non attribute of context. Several tracking technologies have been operating for several years now demonstrating very useful location-aware applications. AT&T Laboratories in Cambridge[1] has been a major contributor by devising two of the most popular location technologies: the *Active Badge* [Want92], infrared room-scale resolution, and *Active Bat* [Ward98], 3-D ultrasonic fine grain, indoor location systems. Both systems require people and objects to be located to be attached an electronic tag that transmits a unique identifier via either an infrared or ultrasound interface to a network of sensors in the walls or ceilings of a building. A Location Server then polls the information from the sensors and makes it available to applications. Examples of interesting applications that have been produced using these technologies are:

- *VNC* [Richardson98] *teleporting* [Richardson94], moving the user desktop to her new location
- *Telephone call routing* to the phone nearest to the addressee
- *Walk through videophone*, which automatically selects streams from a range of cameras to maintain an image of a nomadic user.

The above mentioned two location technologies, despite being probably the most reliable and useful existing indoor tracking technologies, present some

[1] Formerly known as ORL

inconveniences. The tags they use need of battery power and are not cheap, and the infrastructure required, a network of sensors in the walls or ceilings of a building, is complex to install and maintain and also expensive. This line of argument led us to devise an alternative sensor technology whose ultimate goal is to provide a better trade-off between the price and flexibility of the technology and the accuracy of the location data provided.

TRIP (*Target Recognition using Image Processing*) is the name of the new sensor technology proposed that by means of commonplace Image Processing and Computer Vision algorithms processes video frames captured by cameras and recognises 2D circular barcodes in them. The information inferred is the approximate location, orientation and identifier of the target barcodes sighted.

1.1. Research Motivation

As Section 2 will further show, sensor technologies employed in the area of indoor location-aware computing involve complex and expensive special purpose designed sensors to be deployed and the use of electronic battery-powered mobile positioning devices that transmit a signal via an infrared, ultrasound or radio wireless interface. This research tries to demonstrate that similar usability levels as obtained with previous location technologies can also be achieved by using existing off-the-self technology in an easier and more cost-effective way.

1.2. Research Statement

The novel vision-based sensor proposed uses commonly available technology, conventional video cameras and PC processing power, with the aim of obtaining the identity and accurate 3D location of passive cheaply printed barcode tags in the field of view. The main stake of this work is to demonstrate the potential of this sensor technology to be applied as a tracking technology in a similar way other predecessor technologies where used, but still aiming a better trade-off between the scalability, infrastructure complexity and price of the system. This work also tries to explore different application areas opened by the peculiar characteristics of this sensor system.

1.3. Report Outline

Section 2 overviews the state of the art in sentient computing research. Location technologies, architectures to manage contextual data and interesting context-aware applications are reported. Section 3 describes the design principles of the TRIP vision-based sensor technology. Section 4 provides an evaluation of the performance and accuracy figures that are provided by the system. Section 5 provides an insight into the event-based Distributed Component architecture devised to manipulate and distribute the sensorial data provided by TRIP. Section 6 performs a critical evaluation of the architecture designed, remarks its limitations, and proposes future improvements. Section 7 summarises future work to be carried out from now on and produces a schedule with an estimation of landmarks and time deadlines to be achieved. Finally, section 8 draws some conclusions.

2. Related Work

This section reviews previous sentient computing research found in the literature with a special emphasis on the more reduced domain of location-aware computing. Firstly, sensor technologies to determine the location of entities are examined. Then software architectures designed to efficiently manage context data collected from sensors are analysed. Finally innovative applications that show the potential of this research area are briefly overviewed.

2.1. Location Technologies Overview

Ward [Ward98] stated that a good location technology should provide fine-grain spatial information at a high update rate, be unobtrusive (small, lightweight and wireless), be scalable by allowing the location of many objects in a wide area, low-powered, software-supported and low cost. Different technologies have been devised in the last ten years satisfying these requirements in different form. In what follows a brief description of them is given categorised by their underlying communication technology.

2.1.1. Infrared-based Technologies

2.1.1.1. Active Badge

The Active Badge System [Want92] pioneered the research in indoor location systems and is still a point of reference for any new indoor location technology attempt. An Active Badge is a small device worn by personnel that transmits a globally unique code using an infrared data link every 10 seconds. A simpler version of the Active Badge is also designed for tagging equipment. Each office within a building is equipped with one or more networked sensors that detect badge transmissions. The location of the badge wearer is determined on the basis of the spatial region where the detecting sensors are contained, since infrared signals do not travel through room walls. The personnel badges contain two pushbuttons that enable their use as ubiquitous signalling devices. They also have a receive capability and can interpret a range of messages. Their small speaker and two visible LEDs constitute a basic paging facility. The resolution of this location system can be improved from *room-scale* to *desk-scale* granularity when a hybrid radio/infrared scheme is used as it was proposed at [Harter94].

2.1.1.2. PARCTAB

The PARCTAB [Want95] experiment is widely cited as the founding work in the field of Ubiquitous Computing[2] [Weiser93]. This project main aim was to provide users with the ability to access computing resources in an un-tethered mobile way. A

[2] Ubiquitous Computing is the method of enhancing computer use by making many computers available throughout the physical environment, but making them invisible to the user.

PARCTAB is a handheld dumb terminal with a 128x64 pixels touch sensitive display, three buttons and a speaker, that uses an infrared-based cellular network for communication. In order for PARCTAB to be used in a building, each room must be equipped with an infrared transceiver, similarly to the Active Badge case, which handles communication with all PARCTabs in the room. PARCTabs act as thin display clients for applications running on fixed machines in the LAN and as active badges emitting infrared signals so that room transceivers know where they are. Objects are located to room scale resolution identically to the Active Badge case.

2.1.1.3. Smart Badge

A Smart Badge [Beadle98] is like an Active Badge but with a collection of sensors and actuators and re-programmability added. The attached sensors measure environmental factors such as temperature, humidity, ambient light level, orientation and sound. The data collected from the sensors together with the unique identifier assigned to each Smart Badge are broadcast periodically across an infrared interface to networked sensors placed around a building. Smart Badges also have an output port to which computing devices can be attached, and to which data can be sent from the fixed sensor network. The main motivation of this active tag is to extend the application scope of the Active Badge, from location-aware computing to the broader sentient computing area.

2.1.1.4. Locust Swarm

The Locust Swarm [Kirsch97] infrared-based system provides location information and messaging capability without the need of battery-powered tags and a network of sensors. It takes into account the privacy concerns associated with active badges by giving the user unique control of the location information and its release to the network. The Locust is dependent on a solar cell to provide its power and thus normally is placed in the grills beneath overhead fluorescent lights. Upon power-up the Locust begins broadcasting its location information. A user's wearable computer can listen to this broadcast and decide whether or not to announce its location across the infrared link. Additionally a wearable computer user can transmit to the Locust an annotation that wants to attach to that location. The Locust stores that annotation and then interleaves broadcasting its location with the stored annotations.

In [Starner97] the position information provided by the Locust Swarm is combined with a wearable computer's see through display and vision-based simple target[3] recognition capability to augment user's view of the environment with physically based hyperlinks. Once a tagged object is uniquely identified, the annotation system of the Locust is used to add overlying text, graphics, or video on top of the object's view.

[3] The tags employed consist of two red squares bounding a pattern of green squares representing a binary number unique to an object.

2.1.2. Ultrasound-based Technologies

2.1.2.1. Active Bat

The Active Bat [Ward97] ultrasound-based indoor location system reports objects' position co-ordinates in a frame of reference rather than the space container within which an object is found as occurs with infrared-based technologies. Small battery-powered wireless units called *bats* are again attached to equipment and carried by personnel. A bat consists of a radio transceiver, controlling logic and an ultrasonic transducer, containing a 16-bit globally unique identifier. Ultrasound receiver units at known points on the ceiling of rooms are networked. A base station periodically transmits a radio message containing a single identifier, causing the corresponding Bat to emit a short pulse of ultrasound. Simultaneously, the ultrasound receivers in the room covered by the base station are reset via the wired network. Receivers monitor the incoming ultrasound and record its time of arrival. Using the speed of sound in air, Bat-receiver distances are then calculated. When distances from the Bat to three or more non-collinear receivers are found, its position in 3D space is determined using a multilateration process.

The Active Bat system's tracking rate is argued to be around 50 Hz, being 95% of the bat readings within 9cm of their true positions. The battery lifetime oscillates in the range 2-4 months. Bats have also input and output (two push buttons) facilities that take advantage of the bi-directional radio link.

2.1.3. Radio-based Technologies

2.1.3.1. RFID tags

A Radio Frequency Identification (RFID) tag is a microprocessor, transmitter, and induction power pickup loop. When irradiated with an electromagnetic signal of the correct frequency the usually battery-less RFID tag charges and then transmits its identity number on a different frequency to a receiver. Operating ranges are usually located within a few meters of a radio interrogator unit. The TIRIS [TIRIS98] tag is a good example of these battery-less RFID tags

The 3D-iD [PINPOINT98] indoor radio-location system based on L3RF (Low range, Long life, Low cost Radio Frequency) tags is argued to read signals from distances up to 30 meters through walls, no line of sight being required. These tags receive a low power 2.4 GHz spread spectrum radio signal from a cell controller and respond at defined intervals with 5.8 GHz signals that include identification data. A cell controller network continually tracks 3D-iD signals. Each cell controller can consist of up to 16 antennas that receive signals back from the 3D-iD tags. By calculating the round trip times of signals detected by its antennas, a 3D-iD cell controller can identify the location of the tag to an accuracy of around 1m. 3D-iD unique L3RF-based tags operate around one year without battery replacement.

2.1.3.2. GPS

The Global Positioning System (GPS) [Dana98] is an outdoor positioning system created by the US Department of Defence. It is based around 24 satellites in Earth orbit that transmit spread-spectrum radio signals, allowing a receiver anywhere on the globe to be located to within 100m horizontally and 156m vertically. The system cannot be used indoors, because the frequencies at which the satellites transmit signals do not penetrate buildings. The satellites emit two distinct signal types: CA (*Coarse Acquisition*) and PPS (*Precise Positioning System*). CA coded signals can give 15 metre RMS (Root Mean Square) accuracy. However, the US Defence Department introduced a random error into the system, known as *Selective Availability (SA)*. This means that satellites will randomly give out an error signal, thus degrading the accuracy of the signals to around 100 metres. PPS is only available to licensed, mainly military, users and can give accuracy below 1 metre accuracy. Differential GPS can be used to remove the inaccuracies introduced by SA. GPS correction information can be broadcast from another receiver at a known location to an optional radio beacon receiver attached to a GPS unit. A good overview of other outdoor positioning technologies is given at [Azuma99].

2.1.4. Vision-based Technologies

2.1.4.1. Pfinder

The Pfinder (*person finder*) [Wren97] tracker applies sophisticated computer vision techniques to recognise the presence of a user in the environment without this needing to wear any special marker. Motion in video images is used to identify the presence of a user and subsequently the system uses a statistical model of colour and shape to obtain a 2D representation of head and hands in a wide range of viewing conditions. Unfortunately, many objects of interest that wish to be located (e.g. workstations) are less mobile and distinctive than people. The task of determining the identity of the person tracked by the system is in addition very hard. To make the vision task tractable Pfinder expects the scene to be significantly less dynamic than the user and that only one user stays in the recognition space. Pfinder has been used to control an interface by gesture and as input for very low bandwidth telepresence applications.

2.1.4.2. SONY Navicam

Rekimoto et al [Rekimot95] proposed a method to identify real world objects and estimate their position and orientation using a combination of visual markers and a video camera. Their approach was to build computer augmented environments [Azuma97] using a situation aware portable device, called *NaviCam* (NAVIgation CAMera*)*. The SONY Navicam is a portable computer with a small CCD video camera to sense real world situations. This system provides the user with an augmented view of the real world by overlaying context sensitive information generated by a computer. Navicam uses a miniature gyro sensor to determine the orientation of the device and possesses a vision-based ID recognition capability to detect the position of the device and real world tagged objects in the field of view.

Initially Rekimoto employed colour-codes containing a sequence of 4 colour stripes (red or blue) to represent an entity ID. Given that the number of possible identifiers (2^4) was limited, a refinement of this barcode technology named Matrix is later exposed at [Rekimoto98]. This new method utilises a 2D matrix marker, a square shaped black and white barcode design which enables to tag a larger number of objects (2^{16} different codes). By analysing the distortion of the rectangular shape of the Matrix code frame, the system estimates the position and orientation of the video camera and determines the transformation matrix between the real world points and the image points. This transformation matrix is then employed to correctly register computer-synthesised information on the real world image. The authors argue an average screen update rate of around 15 Hz by using the processing power of a connected high performance SGI O2 workstation (MIPS-R10000 175 MHz). The maximum distance from the 2D 5cmx5cm size Matrix patterns to the camera for the correct code recognition and information registration is approximately 1 metre.

2.2. Management of Context Information

Tracking systems operating within an indoor environment require of a software architecture that manages the location attribute of potentially a big number of entities and that provides interfaces for their query by applications. This software architecture may additionally capture other factors of the environment such as network activity or sound level to draw a more accurate picture of the current situation. In what follows several software architectures proposed for the management of contextual data are overviewed.

2.2.1. The Active Badge Distributed Location Service

Harter and Hopper [Harter94] proposed a scalable Location Service to manage the indoor location data provided by Active Badges. The centralised Location Server they designed maintains a cache of the last piece of location information for every badge detected. The location unit kept for each badge consists of a badge address, a location and a time-stamp. Interfaces were provided for clients to invoke queries about badge locations, and about location contents. These interfaces are suitable for clients that start, interact with the system and terminate. For long living clients another interface to specify a filter and a callback registration was provided. In this way the Location Service delivers only changes in location reducing significantly the network traffic. A Name Server that provides applications with lookups by name or by address for badges, equipment, locations and domains was also designed. ANSA [ANSA90] was the distributed applications platform employed in its implementation.

2.2.2. The Active Map Location Service

Schilit et al [Schilit94b] described a centralised Active Map Server (AMS) that handles updates and queries over large regions of space and is also able to handle peak loads that can occur when everyone in a region is moving around. An active map consists on a hierarchy of locations with a containment relation, e.g. rooms are

contained in buildings and buildings are contained in a region. Clients of the active map service publish information about objects at a particular location and/or submit queries to obtain information about other published objects locations. Clients can also subscribe to some queries and the AMS sends information as it changes over time.

The Active Map Server is by definition a very similar design to the previous Location Service described. However, it differs on the way location-based messages are disseminated to interested clients. The simplest implementation of the service involves a remote procedure call (RPC) per client interested in an object location update. However, its authors observed most of the AMS load generated during overloaded situations is due to sending the same update message, over and over again, to many different subscribers. The AMS recognises when multiple clients are specifying the same subscription query, by enforcing clients to use given query templates, and employs a *multicast* channel to service the update traffic for that query, requiring only a procedure call per channel. Similarly, when different queries result in updates to the same set of clients a multicast channel is also shared. In essence, their approach uses large numbers of multicast groups in order to keep client filtering overhead and slow communication link loads to a minimum.

2.2.3. The Situated Computing Service (SitComp)

Hull et al [Hull97] observed that previous Location Services [Harter94] [Schilit94b] based their focus on location data collection and distribution but left the sensor data interpretation task to applications. Moreover, these services were intrinsically tied to their underlying tracking technologies, namely Active Badge and PARCTAB respectively. To overcome these limitations they defined a Situated Computing Service (*SitComp*) that interprets sensor data coming from different context-sensing technologies, not only location technologies, and provides context-aware information at an appropriate level of abstraction for applications. Query and notification interfaces are provided to communicate the current situation to applications.

The SitComp service software architecture is composed of a dynamic network of connections between sensors, interpreters, and situated applications. As sensor data flows up through this network, it is combined and abstracted by interpretation layers until the dataflow impacts a level of abstraction exposed via the service's API and an appropriate event is posted to interested applications. All entities in the network join by registering as producers or consumers of situational information. Data fusion and abstraction is achieved by enforcing standard formats for situational dimensions, and ensuring that the output of all entities sourcing a dimension are routed to an appropriate interpreter.

2.2.4. SPIRIT (Spatially Indexed Resource Identification and Tracking)

The SPIRIT [Adly97] project goal is to support mobile users in an office environment that move around without undue degradation in the computing and communications resources available to them. To achieve such purpose information about the environment is gathered from a range of sensor sources, including the Active Badge, the Active Bat and telemetry software monitors for keyboard, CPU, disk and network

activity. The resulting data is combined with static data about the building, people and equipment, to create a detailed model of the environment [Steggles98]. This model sets up the types, names, capabilities and properties of all entities (people, computers, telephones, etc.). The software counterparts of real-world entities are implemented as persistent distributed objects using CORBA and Oracle 7 database. These persistent objects provide information to mobile applications via query and callback interfaces and are accessed via a proxy server.

SPIRIT noted the interest of location-aware applications for relative spatial facts rather than absolute ones, e.g. the Active Bat system provides the fact 'the person is at x, y, z, facing in direction θ' whereas applications are interested in 'the person is standing in front of the workstation'. To address this issue it defines a real-time *Spatial Monitoring Service* [Harter99] that expresses relative spatial facts about objects in terms of geometric containment and overlapping relationships between spaces associated with those objects. Location events generated by object movements are used as input for an indexing system, which calculates all containment and overlapping relationships and broadcasts them to registered applications using a performance tuned event service.

2.2.5. The Context Architecture

The *context architecture* [Dey99a] infrastructure aims to separate context sensing from context-use and to make context-aware application development as ease as GUI programming. It is constituted of three different types of components: *widgets*, *servers* and *interpreters*. *Context widgets* are responsible for acquiring a certain type of context information and make it available to applications in a generic manner, regardless of how is actually sensed. Applications can either query the state of a widget or register specifying the context notifications they will be interested on. *Context servers* gather the context about an entity from the available context widgets and aggregate it acting as a context-widget proxy for the final context-aware applications. *Context interpreters* are used to abstract or interpret context. For example, a context widget embodying a GPS receiver may provide location context in the form of latitude and longitude, but an application may require the location in the form of a street name.

Context components are instantiated and executed independently of each other in separate threads and/or on separate computing devices, being the communication among applications and components supported by HTTP. The main limitation of this architecture is that for an application to use a widget, server, or interpreter, it must know both the hostname and port the component is being executed on. In addition, this system architecture concentrates on easing application development but doesn't provide an event service that addresses information dissemination scalability issues.

2.2.6. The stick-e note infrastructure

Brown et al [Brown97] noted that context-aware applications implementation is complicated and requires the skills of highly qualified systems programmers. To overcome this situation, they proposed a new infrastructure, the *stick-e note*

[Brown96], that makes the creation of *discrete* context-aware applications as easy as producing a Web document. In discrete context-aware applications, separate pieces of information are attached to individual contexts (rooms, time ranges, being with certain people), that are triggered when the user enters those contexts. This infrastructure is targeted to a scenario in which the mobile user carries a Personal Digital Assistant (PDA) with environmental sensors attached (location, orientation, time of day, temperature*)*.

A stick-e note [Brown96] is an electronic equivalent of a Post-it note associated to a particular context. The Standard Generic Markup Language (SGML) is chosen to make easy their exchange and publish process. A repository with stick-e notes resides either in the PDA employed or in a backbone network server from which using a wireless link, e.g. a mobile phone with Short Message Service facilities (SMS), contexts and notes can be transferred. Authors create notes associated to a given context and a general-purpose triggering engine activates these when they match the user's present context

The main criticism of this infrastructure is that the mandatory use of notes as clients of context information makes difficult to retrofit an existing application with context sensing or even build an application that modifies its behaviour in response to a changing environment.

2.3. Application Areas

The first catalogued context-aware applications were produced using Active Badge's location data. In the first paper describing this technology Want et al [Want92] describe a telephone call routing application that permits the redirection of a telephone call to the closest phone to an individual. Harter and Hopper [Harter94] used Active Badge location data to provide hands free access control to workstations and doors and to define a "nearest-printer" service offered to users of portable computers that automatically reconfigures the print command output to the closest printer.

The Active Badge System has also been used for *mobilising* user *applications*. The Teleporting System [Richardson94] used the location information provided by personnel and equipment Active Badges and the control inputs resulting from pressing Active Badge's buttons to reallocate a user's desktop to her new location. Bates et al [Bates96] described a framework for building location-oriented multimedia applications that enables multimedia objects to follow the user.

Schilit et al [Schilit94a] employed the PARCTAB device to implement a variety of applications involving automatic contextual reconfiguration and context triggered actions. A multi-user drawing program was built to provide a virtual whiteboard for a room causing an automatic binding between a mobile host entering a room and the virtual whiteboard. A contextual reminder application was produced that permits a description of the situation for when a remainder should occur. When the context specified is matched a remainder note is triggered. Lamming et al [Lamming94] also used PARCTAB technology in the implementation of their Forget-me-not system, a

context-aware based retrieval application whose function is to act as a human memory prosthesis. This project aims to have a large number of sensors around the workplace in order to capture as much as possible about the user's working life: what rooms they were in, who they were with, what communications they sent and received, and so on. This information is then available whenever the user wants to recall past situations. This project demonstrates that user's context can itself provide a valuable key for indexing information automatically.

The Smart Badge [Beadle98] technology has been used to implement a Smart Hospital application that aims to improve information flow in a hospital. A central computer stores hospital records and provides authentication and access control services. Patients are equipped with Smart Badges that report their current location, have a panic button, and monitor temperature, respiration and heart rate. The doctor carries a Smart Badge attached to a wireless communication enabled PDA where co-located patient records are automatically displayed. Results of each examination performed to the patient are inputted by the doctor into her PDA and made available through a wireless link to the central Smart Hospital Server. Doctors' Smart Badges' input is also used to provide access to restricted areas.

Rekimoto [Rekimoto95] has applied his barcode based identification technology to produce applications in the domain of Augmented Reality. Sony NaviCam's barcode recognition capability has been used among other applications to augment a museum's view with personalised computer synthesised information according to the user's age, knowledge level or preferred language.

The Active Bat System design motivation was to enable new kinds of location-aware applications only possible with a more fine-grained location and orientation resolution. The described Teleporting System [Richardson94] presented the limitation that when a user clicked her Active Badge button her desktop was teleported to one of the several screens available in the containment room, but not necessarily to the closest one. [Harter99] explained how the SPIRIT system [Steggles98] combines Active Bat precise location and orientation with resource monitoring information to allow the redirection of a user desktop to the closest non-utilised display. Likewise, Ward [Ward98] described a walk-through videophone application that enables a nomadic user to carry on her videoconference by monitoring her location and orientation accurately in relation to the available video and sound input and output resources.

The stick-e notes [Brown96] infrastructure has been used to produce PDA-based context-aware applications. [Brown97] described an application that uses stick-e notes to cover paging requests. Somebody looking for a book she can not find can create a stick-e note expressing her wish to obtain it. Whenever, somebody comes across with that item, a paging message is triggered indicating that somebody else is looking for that book. Pascoe [Pascoe98] described an application of stick-e notes technology in an ecological fieldwork. An ecologist was provided with a PDA with an attached GPS unit and a stick-e note infrastructure based application to assist her in the observation and data collection task to investigate the feeding behaviour of giraffes. The context associated to each note was automatically captured by the system (time and location) letting the user concentrate in her observational tasks. Here information was authored in a particular context rather than presented in a particular context.

Dey et al [Dey99b] have used their *context architecture* to create a *Context Assistant*. This application tries to help conference attendees decide which activities to attend, to provide awareness of the activities of colleagues, to assist users in taking notes on presentations and to aid in the retrieval of conference information after the conference concludes. It uses a wide variety of contextual data: time, identity, location and activity. Furthermore, it combines in one application most of the features provided by other context-aware applications: (1) it presents information and services to the user (conference timetable and what colleagues are attending to what conference); (2) automatically executes services (it automatically updates the current slide in the user PDA); and (3) tags context to information for later retrieval (notes made by the user are augmented with contextual information). Conference attendees execute the application in PDAs provided by the conference organisation with attached 3D-iD RF-tags to obtain location. Context based retrieval is also possible from attendee's home location to revise the material exposed in the conference.

3. TRIP: a vision-based sensor technology

TRIP (Target Recognition using Image Processing) is a novel vision-based location sensor that uses the combination of visual markers, 2-D circular barcode tags (see Figure 1), and conventional video cameras to automatically identify and locate tagged real word objects in the field of view. Video frames obtained from cameras are processed using commonplace Image Processing and Computer Vision algorithms, optimised to reduce the computational cost to a minimum, to obtain the identifier and approximate location of *TRIP tags* or targets sighted.

TRIP constitutes a very cheap sensor technology. Its 2-D barcode printed tags are easily obtained with a simple target POSTSCRIPT code generating script; it only involves ink and printer usage costs. The infrastructure needed is also cheap. TRIP is primarily software based; the only hardware required being a source of digital pictures plus some CPU processing power borrowed from conventional PCs. The *TRIP Video Filter* software in charge of undertaking the target recognition process has been written in C++.

The main features of this new sensor system are:

- *Tagged.*
  It associates a unique, practically unobtrusive, small barcode passive tag with each sensed object. This approach confronts other conventional identification or location technologies that require of electronic battery-powered tags and some complicated wireless technology to transmit its identity to a detector.

- *Directly interpreted*
  TRIP measures its barcodes' properties of interest: the central bull's-eye location and encoded identifier directly without requiring any human perceptual talents or complicated AI techniques. The employed *target* special design makes the recognition process simpler, faster and accurate. Still the passive tags chosen can be easily attached to large number of objects to obtain their identify and location attributes.

- *Infrastructural?*
  TRIP is easily usable in a standalone fashion with just a camera and targets distributed in the environment, providing space for applications in the areas of Augmented Reality and Wearable Computing as in [Rekimoto98]. Alternatively, it could be fed from a number of known networked cameras in a building, to serve as an indoor location system alike to the Active Badge system. Most of the research done during this year has been focused on the application of TRIP for the second scenario, however forthcoming work will also explore the first one.

3.1. A 2-D barcode as a Location Device

1-D Barcodes have been used for many years as a way of identifying objects, serving as a key for a database. 2-D barcodes [BARCODE98] were proposed more recently for the following two purposes not addressed by 1-D ones:

1. Allow the barcode be a portable database rather that just a database key.
2. Remove the vertical redundancy of the conventional 1-D barcode, enabling the tagging of objects where only a small amount of space is available.

TRIP proposes a completely different application domain for 2-D barcodes by employing a barcode as a *mobile positioning device*. The design of this barcode, shown in Figure 1, was guided by the requirement that these targets should be located at the furthest possible distance from the source of digital images. Its main features are:

- A TRIP target is a 2D black and white circular barcode representing a *ternary* number.
- A circular *bull's-eye* makes the identification process easier due to its properties of invariance to rotation and perspective, and high contrast.
- A couple of code rings around the *bull's-eye* encode its identifier. Each ring code provides 16 bits of information that are read in anticlockwise fashion:
  - The 1st sector (bit 1 of ring code 1 and 2) or *synchronisation sector* special and elsewhere impossible configuration serves to distinguish the beginning of the code.
  - The 2nd and 3rd sectors are used to implement even parity error checking.
  - The remaining 13 sectors correspond to the ternary digits representing the barcode identifier or *TRIPcode*.
    - The number of possible identifiers is: $3^{13} = 1{,}594{,}323 \approx 2^{20}$ valid codes

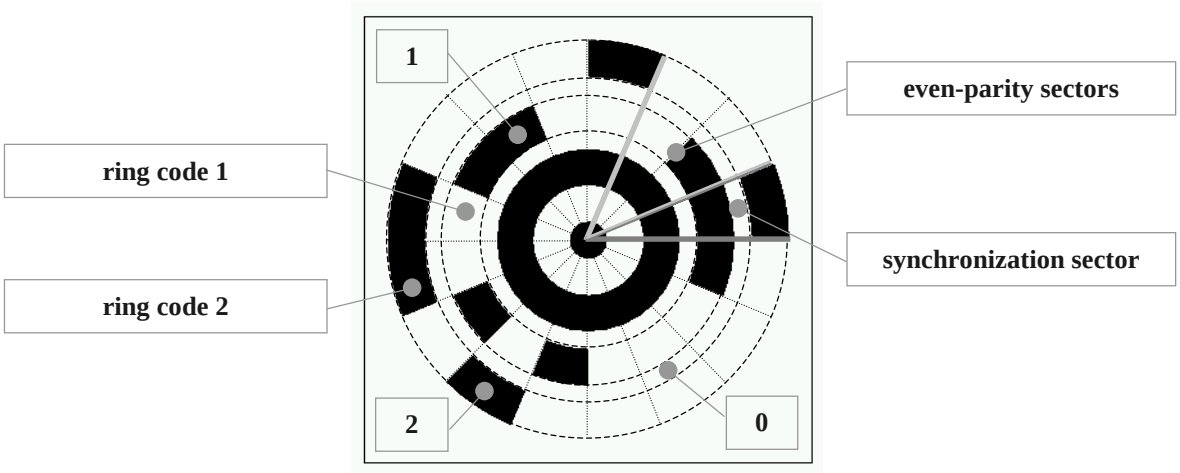

Figure 1: Target representing 1160407 ID.

3.1.1. TRIP Target Design Issues

Reducing the sector width from the current 22.5° to 15°, the range of possible identifiers would be increased. This modification would give the technology a range of identifiers big enough to consider *Error Correction Codes* schemes such as Hamming [Tanenbaum96] where the usage of some redundant bits allow the correction of 1 and the detection of 2 bit errors. Code encoding redundancy policies could also be applied to enable the ID recognition of targets suffering from partial occlusion.

A TRIPcode represents, in principle, an unstructured identifier. In order to allow the efficient utilisation of the barcode identifiers addressing space, a centralised TRIP Code Granting Service is required. This service would allocate new unique TRIPcodes and assign TRIPcode ranges to different categories. In this way, the potential duplication and misuse of barcodes would be avoided and codes corresponding to a certain category would be distinguished by a common prefix. Furthermore, a code by itself lacks any meaning for final applications. It is necessary to establish a correspondence between codes and attributes associated to them, i.e. the creation of a Directory Service. These two requirements have been addressed by the implementation of a *TRIP Directory Server*. Section 5.2.4 will provide a deeper insight into this software component.

3.2. Target Recognition Process

The *Target Recognition Process* takes the original raw video data captured by a Frame Grabber and executes a set of image processing and computer vision stages over it. This process subsequently eliminates all the non-relevant information to the identification and location of printed TRIP targets in the field of view. Figure 2 shows schematically the video filtering process undertaken to recognise TRIP tags in a video frame. In what follows the TRIP Video Filter processing stages are described:

3.2.1. Image Acquisition (Stage 0)

A video camera attached to a computer provides grey scale digital images through a TV video card into a PC Linux machine running a *Frame Grabber* program.

3.2.2. Image Contrast Enhancement (Stage 1)

In this stage the previously captured grey scale video image's pixels are examined and transformed into either black or white intensity values, using the *Adaptive Thresholding* method described by Wellner [Wellner93]. This method varies the threshold value employed as criteria to transform a pixel into black or white value taking into consideration the background illumination of each pixel. As result of this stage the effects of shadowing in the scene of view are removed and the contrast of the image is enhanced. This processing is needed because the video frame sources can be diverse, being likely sources even existing security cameras with low-resolution and contrast level. This stage makes TRIP software very robust under variable lighting conditions.

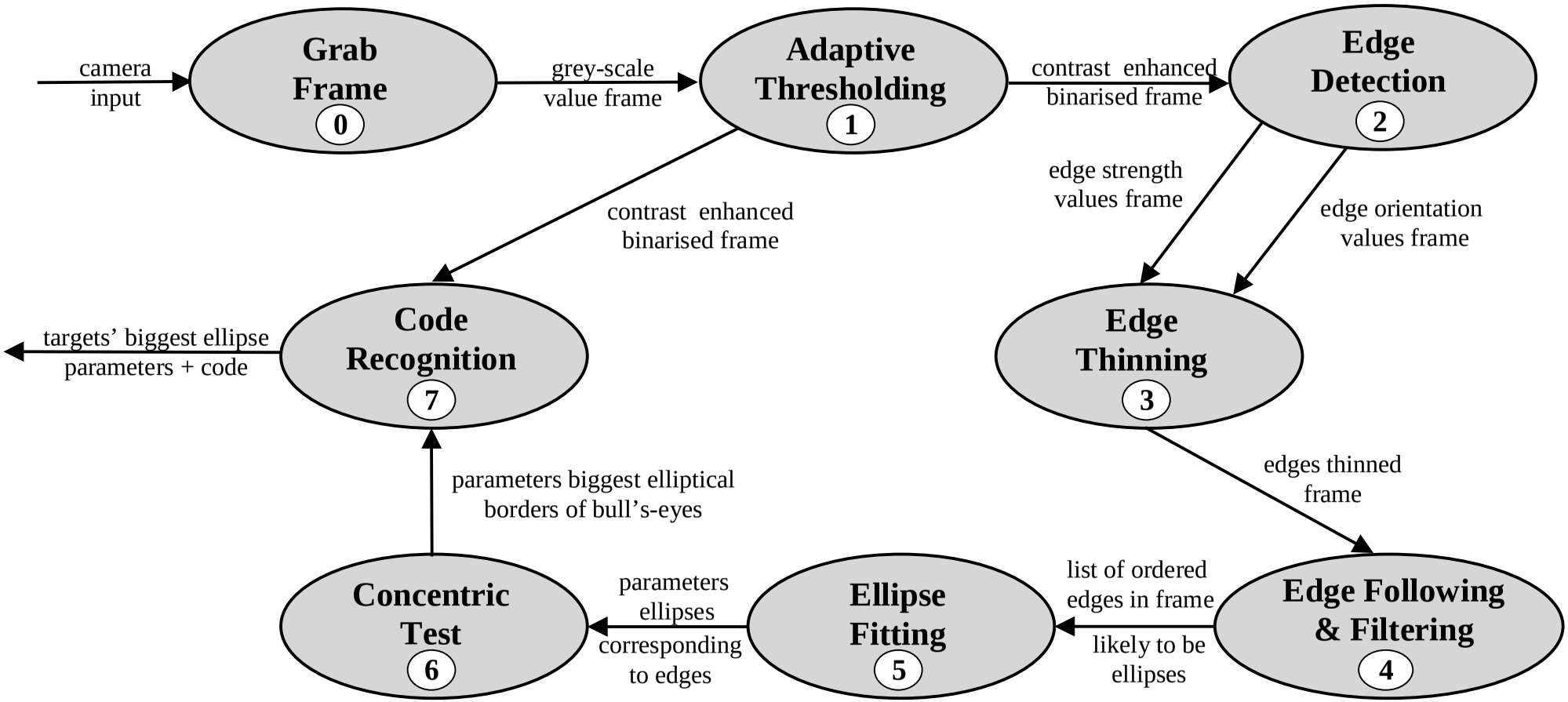

Figure 2: Target Recognition Process

3.2.3. Edge Enhancement (Stage 2)

The pixels at or around which the image values undergo a sharp variation, i.e. its edges, are identified. This stage corresponds to an *Edge Passing Filter* that applies the geometric interpretation of the gradient, the measure of change in a function, to an image, expressed as the rate of change of the grey levels in it. This rate of change is large near an edge and small in constant areas. The gradient operator approximation employed is implemented numerically applying the following Edge Detection masks to the intensity values of each pixel in the image:

$$[1 \;\; 0.5 \;\; 0 \;\; -0.5 \;\; -1] \text{ and } [1 \;\; 0.5 \;\; 0 \;\; -0.5 \;\; -1]^T$$

Two optimisations are done in order to speed up this stage's CPU consumption time:

1. In the calculation of each edge point gradient strength, the Manhattan Distance operator:
$$d = |a| + |b|$$
instead of the Euclidean Distance operator:
$$d = \sqrt{a^2 + b^2}$$
is used to avoid the expensive computationally square root operation.

2. In the calculation of each edge point gradient orientation, the CPU intensive arctangent trigonometric operation is substituted by the arctangent operator algorithm described by Sedgewick at [Sedgewick94].

3.2.4. Edge Localisation (Stage 3)

A decision is made about which local maxima in the *Edge Filter's* output are edge points and which are just caused by noise. This stage involves [Trucco98] two subtasks:

1. Thinning wide edges to 1 pixel width (*non-maximum suppression*).
2. Establishing the minimum value to categorise local maxima as a true edge (*thresholding*).

3.2.5. Edge Following and Filtering (Stage 4)

All the connected chains of edge points (*edgel*) previously located are followed in a clockwise fashion, producing for each edge tracked a list of ordered point locations.

TRIP tags' circular bull's-eyes will be observed in the captured frame as elliptical due to spatial transformations. The *TRIP Video Filter* designed aims the localisation of these *bull's-eyes*, i.e. at least two '*concentric*'[4] edge ellipses. Consequently, in this

[4] Note that when we say concentric we mean 'approximately concentric', due to the fact that spatial distortions make the ellipses not to be concentric in a truly mathematical sense.

stage a filtering process is applied to every edge tracked, retaining only the ones whose *edgels* plausibly belong to an ellipse. The criteria undertaken is to filter out all non-closed edges whose ratio, between its perimeter in pixel number and the distance between the extreme points is bigger than an empirical value greater than 1, the case of straight lines.

3.2.6. Ellipse Detection (Stage 5)

The previous stage provides edges that are good candidates to define an elliptical shape. The Ellipse Detection phase seeks for each of these edges a conic function representing an ellipse:

$$F(a, x) = a \cdot x = ax^2 + bxy + cy^2 + dx + ey + f = 0$$

where $a = [a\ b\ c\ d\ e\ f]^T$ and $x = [x^2\ xy\ y^2\ x\ y\ 1]^T$, whose choice of parameters best matches the observed locations of the given points in the edge, in the *least squares sense*, i.e. by minimising the sum of squared algebraic distances of the curve to the N edge points given, x_i:

$$\min_a \sum_i^N \left| x_i^T a \right|^2$$

The implementation of this stage was based on the "Direct Least Squares Fitting of Ellipses" method described in [Fitzgibbon96].

3.2.7. Concentric Test (Stage 6)

The ellipse parameters obtained in the previous stage are compared two identify concentric ellipses likely to form candidate targets' *bull's-eyes*.

3.2.8. Code Recognition (Stage 7)

Taking as input the black and white intensity values image resulted from the Adaptive Thresholding stage and the parameters corresponding to the outer ellipse of each bull's-eye candidate obtained in the previous stage, the following two operations are followed to determine a target identifier:

1. *Identify the synchronisation sector.*
   The bull's-eye ellipse of reference is transformed to the unit circle, since the ratios between the radius of the bull's-eye and the code rings circumferences are only known with respect to the TRIP target design. Once this is done the intersection points between an imaginary line drawn through the centre of the target and the two imaginary circumferences going through the middle of ring codes, concentric to the unit circle, are determined. These intersection points are transformed back to the corresponding image location using the inverse transformation to the one employed to convert the reference ellipse to the unit circle. If the intensity values of these points sampled on the output of the *Adaptive Thresholding* stage, correspond simultaneously to the black intensity value in the two ring codes it

means the synchronisation sector has been identified. This process is repeated interactively by rotating the reference line 15° and calculating the new intersection points. If the synchronisation sector can not be found after 12 interactions, it means the candidate bull's-eye is spurious and therefore, rejected. If it was found, the beginning of the synchronisation sector is located sampling points in anti-clockwise sense till no corresponding black points are simultaneously found.

2. *Decode the barcode identifier.*
   After the previous operation is completed, sample points in the middle of each of the 22.5° code sectors are taken following the same transformations. If the black colour intensity value is found in a point belonging to the first ring code, the ternary value in that sector is 1, if in the second is 2, and otherwise is 0. Before the decoding operation is completed an even parity error check is applied. If this test is passed, it is concluded a valid TRIP code has been identified. If not the assumption adopted is that the TRIP target identified was spurious.

The Target Recognition Process final result is a list with each recognised target's identifier and bull's-eye's outer ellipse parameters. The ellipse parameters given are the coordinates of the centre of the ellipse in the frame (x,y), its abscises (a and b) and the tilt of the ellipse with regards to the horizontal abscise of the picture frame. Figure 3, demonstrates the TRIP video filtering operation over an example frame grabbed. Observe the last window displays a small cross over the centre of each TRIP target to show the target recognition has been satisfactory.

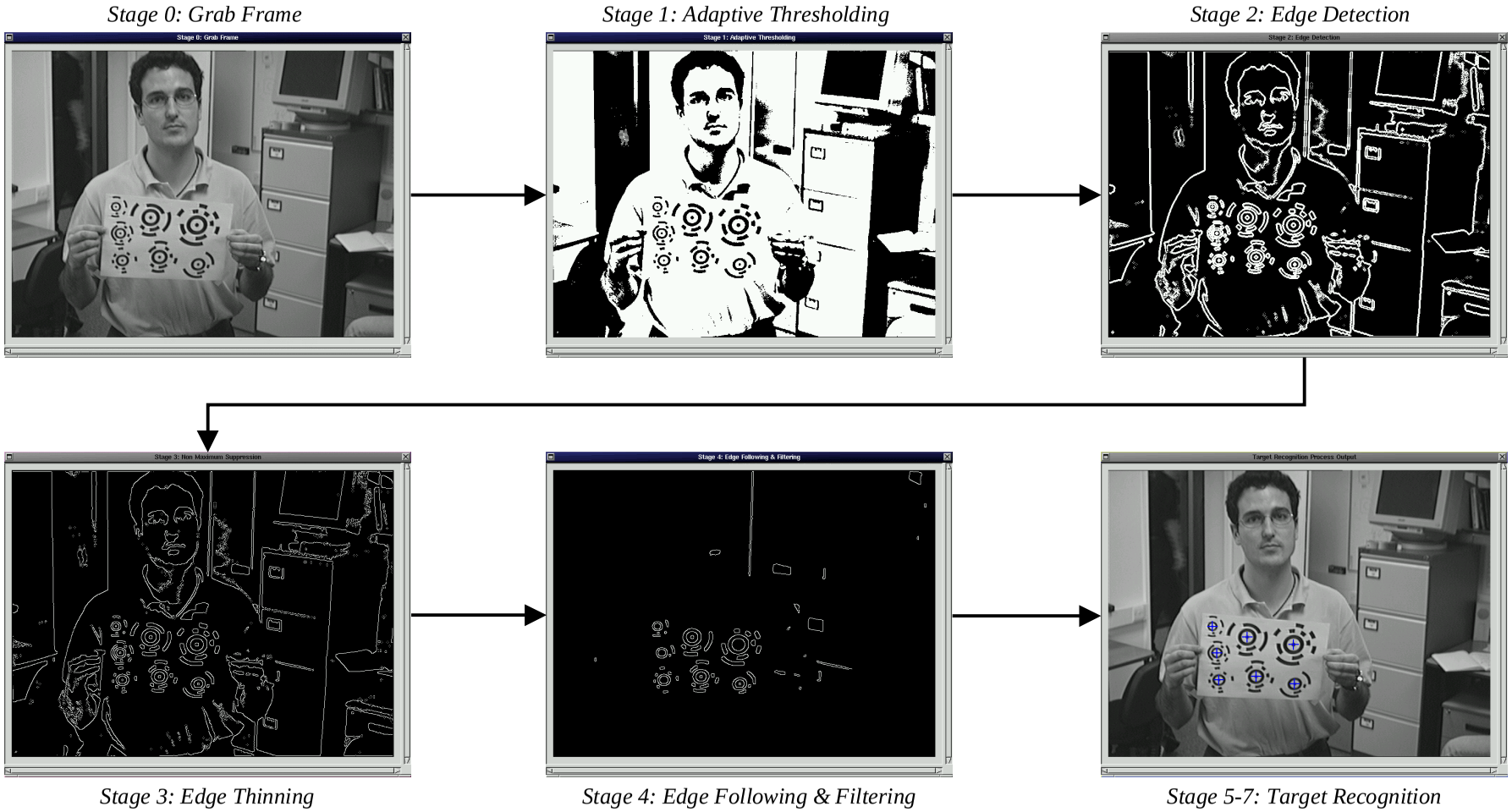

Figure 3: Target Recognition Process Stages Visualisation

4. TRIP Sensor System Evaluation

This section details the performance and accuracy figures of a C++ implementation for the Target Recognition process described in Section 3.2. In addition, it assesses the potential applicability of the sensorial data provided by TRIP and comments its principal limitations.

4.1. TRIP Accuracy and Resolution

In order to evaluate the TRIP robustness and accuracy features an *'eye-test'* was done to the system. The following 4 parameters were studied:

1. *Size* of targets in pixel number as observed in the frame captured.
2. *Tilt* of the camera plane with regards to the target plane.
3. *Greyness* or contrast between the foreground and background of the TRIP barcode, by printing out different TRIP tags with varying degrees of greyness.
4. *Distance* from the target to the camera.

Video frames of 768x576 pixels PAL resolution captured from a cheap auto-zooming analogue camera were analysed. The results obtained concluded a target is spotted with a likelihood of 98-100% whenever the following conditions are fulfilled:

1. The perimeter of the biggest elliptical border of a *bull's-eye* in pixel number is bigger or equal to 25 pixels (approximately 4 or more pixels radius).
2. The tilt of the target plane with respect to the camera plane is less than 70 degrees.
3. The grey level difference between the target foreground and background is bigger than 50%. This factor gives an indication of the robustness of the system under varying lighting conditions.
4. The distance between a target and the camera used is less than 3 meters. More expensive video cameras of higher resolution would enable an increase of this distance.

The TRIP Video Filter software reduces a high bandwidth meaningless input constituted by a video frame into a more meaningful low-bandwidth list of sighted TRIP targets descriptions. A target description is a data structure composed of the target's identifier plus its bull's-eye's outer ellipse parameters. This information can be directly applied to create a containment based indoor location system. Given the camera position is known the location and orientation of the spotted TRIP tag is determined to the granularity of the camera view range. Further location resolution would be achieved by processing images coming from different video cameras located at known positions within a room. If the same TRIP tag is identified within two different source's images, Stereo Vision techniques can be applied to interpret TRIP sensor's output and obtain the 3D Location and orientation of the target. Work so far has been focused on only the first, simpler scenario and therefore the current implementation of TRIP provides only *containment-based* location information. However, means of providing TRIP with higher location resolution features are proposed for future work in section 7.

4.2. TRIP Performance

The C++ implemented TRIP Video Filter software provides a recognition rate of roughly 3Hz. Nearly 3 (768x576 pixels) video frames per second were processed, on a Pentium II 450 MHz with 64Mb RAM running Red Hat 5.2 Linux. The *gprof* UNIX profiling tool was used to study the CPU consumption time of each of the Target Recognition Process stages. This process was run 30 times on the frame shown in Figure 3. Table 1 lists the time and percentage of total time invested in each stage.

As observed in Table 1, stages 1 and 2 (*Adaptive Thresholding* and *Edge Detection*) consume around 78% of the total processing time. Thus, special attention should be paid to their optimisation in future work. The time invested in stage 4 (*Edge Following & Filtering)* is directly proportional to the variable number of *edgels* identified in an image. Remember this process has to go through all these points trying to assign them to an edge. Its CPU time consumption is also quite significant (10%) and should be considered in future optimisations. Stages 5 to 7 (*Ellipse Fitting*, *Concentric Test* and *Code Recognition*) processing time is also variable and differs depending on the number of elliptical edges identified. From them only stage 7 is costly computationally (5% CPU time) and worthwhile optimising.

Stage	% Total Time	Total Time(s)
0) Grab Frame	0	0
1) Adaptive Thresholding	38.715	0.14935484
2) Edge Detection	38.798	0.14967742
3) Edge Thinning	5.603	0.02161290
4) Edge Following & Filtering	10.294	0.03971012
5) Ellipse Fitting	5.936	0.02290321
6) Concentric Test	0	0
7) Code Recognition	0.654	0.00252539
TRIP Software Total Processing Time		0.38578388

Table 1: Target Recognition Process Performance Figures

4.3. TRIP Limitations

The main limitation of TRIP is its line-of-sight requirement between a target and a camera. Several cameras are required to ensure comprehensive tracking coverage of an indoor environment due to TRIP's sensitiveness to occlusions. Partial occlusions where the central bull's-eye can be clearly seen but the barcode rings are partially blocked may be overcome by using redundant code bits as proposed in section 3.1.1. A second inconvenience of the TRIP sensor system is that it requires environments with good lighting conditions. It is impossible to locate a target in dark conditions unless the TRIP tag is printed in a fluorescent material. Finally, other tagging positioning systems such as Active Badge and Bat facilitate a couple of buttons to serve as a ubiquitous control device that obviously are not provided by the TRIP passive tags. Perhaps smaller size special barcodes could be printed out beside the principal TRIPcode that when occluded by the user would simulate control signals.

5. A Distributed Sentient Information Service for TRIP

This section describes the software architecture of a *Sentient Information Service* (SIS) prototyped to manage TRIP sensor data. This service aims to gather sensorial information provided by distributed sensors throughout the environment, not necessarily only TRIP ones, and to provide efficient ways to communicate such information to interested applications. Although, its design has been undertaken with maximum flexibility in mind, in our first prototype we have experimented with sensorial data coming from only one type of sensor, a TRIP sensor placed in a room. Eventually this architecture should assist in the creation of a Location Service for TRIP. The design of SIS intends to fulfil the following functional requirements:

1. *Heterogeneous and Distributed capture.*
   Sentient information must be acquired from multiple distributed and sometimes heterogeneous sensors. For instance, tracking the location of users in an office requires gathering information from multiple location sensors distributed throughout the office. Potentially, TRIP location data could be combined with the results obtained through alternative location technologies such as the Active Badge or even other kinds of sensorial data, such as the temperature obtained from a digital thermometer.

2. *Interpretation*.
   The raw information acquired must be abstracted to make sense to applications. For example, a person's target sighting expressed by TRIP as the combination of the recognised target's geometric details and the camera ID from which the video frame analysed was obtained must be translated into the corresponding person name and location (i.e. room where the detecting camera is).

3. *Real-time Distribution of location data*.
   Changes in the environment must be detected and communicated to applications in real time. The SIS must post events to interested applications whenever some aspects of the current situation change. Moreover it has to provide query interfaces to allow applications to interrogate the current situation.

The proposed Sentient Information Service is modelled as a group of collaborative event-based *Distributed Software Components*, whose goal is to ease the development of *context-aware* applications by hiding the complexity of context-sensing activities and providing appropriate abstraction to the incoming sensor data. In the case of this research our focus has been to benefit from this architecture to implement TRIP-aware applications. This architecture has provided us a way of writing *TRIP-aware applications* in the same *event-driven style* as traditional GUI applications.

The SIS architecture resembles the *context architecture* proposed at [Dey99a] on its stake to ease sentient application development. However, it is also concerned with the efficient dissemination of sensorial events among its components and final applications, similarly to previous research efforts made by [Schilit94b] and [Szymaszek98]. Section 5.2 gives a detailed description of the architecture devised. Before section 5.1 surveys some CORBA distributed technology concepts essential for the understanding of this architecture.

5.1. Overview of CORBA technology

CORBA is a distributed object computing middle-ware standard being defined by the Object Management Group (OMG) [OMG98a]. CORBA is designed to support the development of flexible and reusable distributed services and applications by:

1. *Separating interfaces from remote implementations*. The OMG Interface Definition Language (IDL[5]) is defined to provide a standard interface to systems implemented using different operating systems and implementation languages.

2. *Automating many common network programming tasks* such as object registration, location, and activation; parameter marshalling and de-marshalling; and operation dispatching.

The core component of the CORBA architecture is the *Object Request Broker* (ORB). The ORB allows clients to invoke operations on remote object implementations without concerns for where the object resides, what language the object is written in, the OS/hardware platform, or the type of communication protocols and networks used to interconnect distributed objects. Client and server objects may exist within the same machine or on different machines. An *object reference* is an identifier that uniquely specifies an object within a distributed ORB system.

The CORBA *Common Object Services* (COS) specified at [OMG98b] are architectural models and interfaces that factor out common services for developing distributed applications. In this work the *Naming Service* and specially the *Event Service* are employed.

The *OMG Naming Service* is the principal mechanism for objects on an ORB to locate other objects. The naming service maps humanly recognisable names to object references.

The *OMG Event Service* allows objects to dynamically register or un-register their interest in specific events. Objects generating events don't need to know the interested parties, this all is handled by the Event Service, which enables applications to use a *de-coupled asynchronous communication model* rather than strict client-to-server synchronous requests invocations.

In the OMG Event Service model, *supplier* objects produce events and *consumer* objects receive them. Both suppliers and consumers connect to an *Event Channel* that is an intervening object that is both a supplier and consumer of events. This object allows multiple suppliers to communicate with multiple consumers *asynchronously* and without knowing about each other. It is responsible for supplier and consumer registration, timely and reliable event delivery to registered consumers, and the handling of errors associated with unresponsive consumers.

[5] IDL is a declarative language whose grammar is a subset of C++ with additional keywords to support distributed concepts.

The OMG Event Service provides two models of event delivery: the *push* and the *pull* models. With a *push* model, suppliers take the initiative and push events to the *Event Channel* that then subsequently pushes them to consumers. For the *pull* model, the actions that cause event flow occur in the opposite direction. Event channels not only support both *push* and *pull* models for event delivery but also allow the models to be mixed. Suppliers, consumers and Event Channels handle event data in the form of the IDL *any* type[6], with enables event-based applications to send and receive domain-specific event data without requiring Event Channels to understand them.

5.2. The Sentient Information Service (SIS) Architecture

The Sentient Information Service architecture aims to provide a general and flexible framework that could potentially be used by heterogeneous sensor technologies as long as they support the same event based model. This architecture is composed of a group of distributed CORBA components that communicate one with the other through OMG Event Channels. It uses events as a uniform way of informing other components of activities that have occurred. New components can be integrated into this architecture as event consumers, event suppliers or as event consumers & suppliers simultaneously. Consumer components register with Event Channels that serve their events of interest. Supplier components create new Event Channels, where after registering communicate their own events. 3 component categories are proposed:

1. *Context Generators.*
   They are in charge of acquiring context information. They encapsulate a single sensor or a set of related sensors and the software that acquires raw information from them. The raw information acquired is transferred to *context communicators* in event form, following usually a *push* event communication model.

2. *Context Abstractors.*
   They are seen by applications as *proxy context generators*. *Context abstractors* achieve the separation of concerns between context sensing and application semantics. They consume the raw sentient data provided by context generators, interpret its contents and augment them with static data retrieved from a database, producing enriched contextual events that can directly drive final applications. Sometimes context abstractors need to *correlate* other context abstractors' or context generators' outcomes to generate the contextual data demanded by an application.

3. *Context Communicators.*
   They are the intermediary entities that de-couple the communication among components of the previous two types and final applications. They constitute the glue that enables the heterogeneous software components and applications conforming this architecture to inter-operate and are physically implemented as OMG Event Service Event Channels.

Figure 4 shows the SIS architecture and some potential applications connected to it. Although in the diagram only one context generator is shown, potentially several

[6] The *any* type is useful in dynamic situations because it can represent any possible IDL data type.

context-generators could co-exist as long as they provide an Even Channel where interested parties in their notifications could be registered.

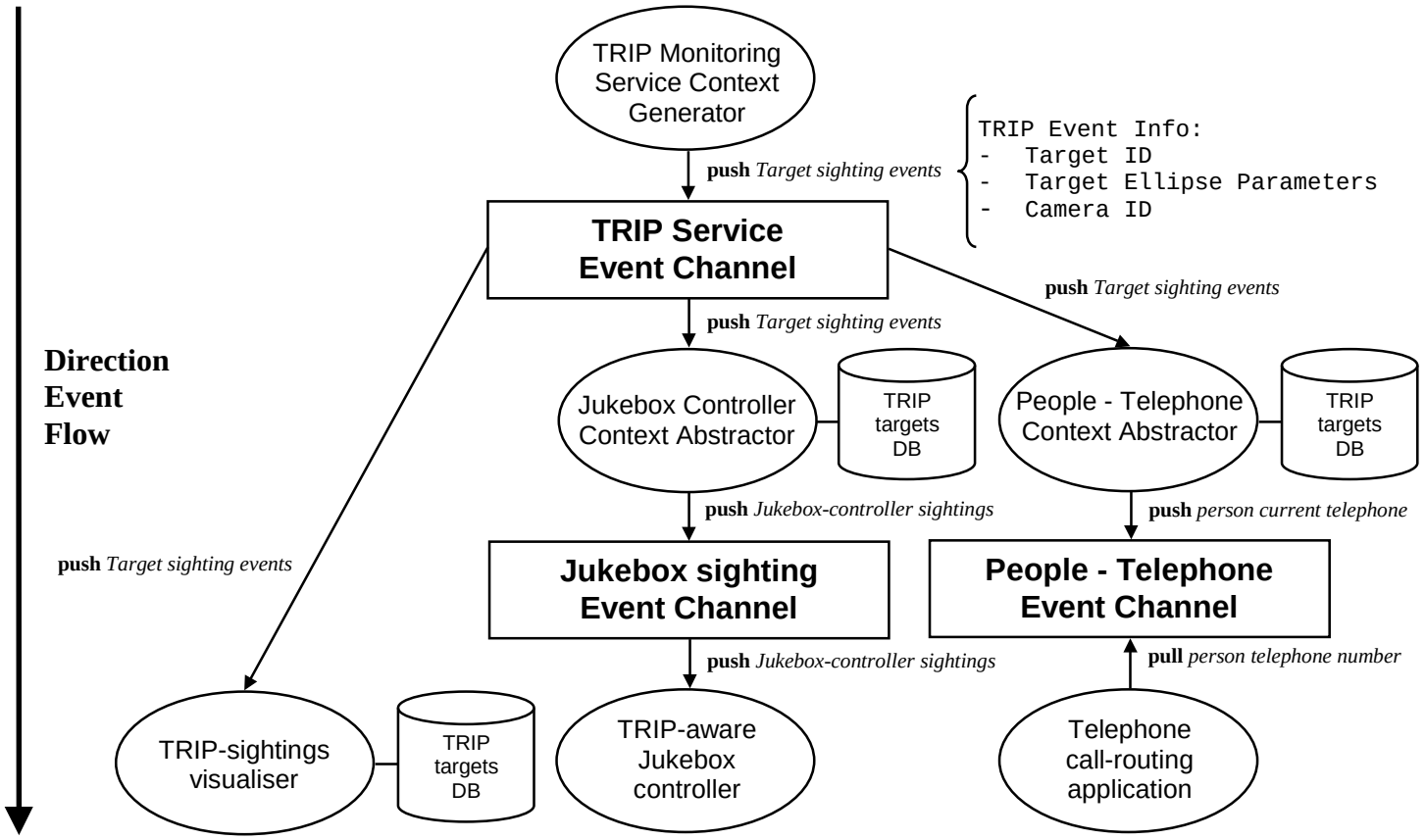

Figure 4: The Sentient Information Service Architecture

5.2.1. The TRIP Monitoring Service Context Generator

The *TRIP Monitoring Service* is a *context generator* component that receives video frames captured from various cameras, processes them using the *TRIP Video Filter* software overviewed in Section 3.2 and pushes TRIP target sighting events into its context communicator *(TRIP Service Event Channel)*. A TRIP target sighting notification contains the ID and geometrical parameters of the barcode target seen and the identifier of the camera from which the frame analysed was captured. Applications and/or context abstractors can then connect to the TRIP Service Event Channel and consume the sightings notified. This component was implemented in C++ using the CORBA 2.0 compliant ORB omniORB2 [Lo99]. It also uses omniORB's COS Name Service and Event Service implementations, named *omniNames* and *omniEvents* respectively.

At its bootstrap the TRIP Monitoring Service component creates the *TRIP Service Event Channel* object, registers to it as *push* supplier and then binds its object reference with the Naming Service under the name *TRIPMonitoring*. Context abstractors and applications can later use this well-known name to obtain the component's object reference and invoke the `getEventChannel` (see Listing 1) remote procedural call to obtain a reference to the *TRIP Service Event Channel*.

Interested parties then register as consumers of it. A *TRIPevent* type (Listing 1) value is communicated each time a TRIP sighting notification takes place.

```
module TRIPMonitor{
  interface TRIP {
    (…)
    // Event interface:
    CosEventChannelAdmin::EventChannel getEventChannel();

    // Event structures:
    struct TRIPevent {
      string code; // ternary representation of the TRIPcode
      paramsEllipse params; // parameters of the outer bull's-eye ellipse (x,y,a,b,θ)
      string cameraID;
    };
    (…)
  };
};
```

Listing 1: TRIP Monitoring Service Event Interfaces in IDL

5.2.2. Sentient Information Service TRIP-aware Context Abstractors

Context abstractors receiving events from the TRIP Monitoring Service undertake the following two tasks:

1. Filter out target sightings that don't correspond to their domain of interpretation.
2. Interpret relevant target sighting events and generate new enhanced events containing the actual sentient information required to drive final applications.

The *Jukebox Controller Context Abstractor* is the only one of the two context abstractors shown in Figure 4 that has already been implemented. At its initialisation stage, it registers with the *TRIP Monitoring Service Event Channel* as push consumer, creates the *Jukebox TRIP sightings Event Channel* where it will push its own events and binds its object reference with the Naming Service to enable clients to contact with it. Its main task is to filter out all target sightings that don't belong to the Jukebox-controller application domain, and from valid target sightings generates Jukebox-controller events augmented with data from a TRIP targets database. A valid target code may represent *music tracks* to be played, people whose *playlist* should be selected or *jukebox control actions* such as play or pause. When the TRIP code representing a song is spotted, the file location of that song will be obtained from the database and pushed as the attribute of a new event of type *song* into the *Jukebox TRIP sightings Event Channel*. When a person's TRIPcode is sighted the file path containing the user playlist will be pushed as attribute of an event of type *playlist*. Finally, if a jukebox action TRIPcode is seen an event of type *jukebox-action* containing as attribute the type of action to carry out will be pushed. Listing 2 shows the IDL types of the events generated by the Jukebox Controller Abstractor.

As it was commented in section 4.1, the information that the current implementation of TRIP provides is sufficient for the deployment of a containment-based location system. In this way, we could imagine a *Location Service Context Abstractor* that would receive notifications from TRIP Monitoring Service Components covering a building, map the camera ID attribute of received events to locations and keep a cache with the last location determined for each entity tagged in the environment. This context abstractor would provide apart from the conventional event interfaces of other

context-abstractors, interfaces for client applications query TRIP wearer locations, *query by name*, or query the objects located in a given room, *query by spatial position*. Section 7 proposes this context abstractor as further work.

```
module JukeboxAbstractor{
  // Event interface:
  CosEventChannelAdmin::EventChannel getEventChannel();
  // Event structures:
  struct playlistEvent {
    string playlistFilePath;
  };
  struct MPEG3TrackEvent {
    string songToPlay;
  };
  struct actionEvent {
    string action;
  };
};
```

Listing 2: Jukebox Controller Context Abstractor Event Interfaces in IDL

5.2.3. The Jukebox controller TRIP-aware Application

Applications can either directly consume raw context information from context generators' Event Channels or more conveniently obtain the enhanced sentient data from context abstractors' Event Channels. Due to the Event Channel's features contextual events can be acquired following either a *push* or *pull* event communication model. An application only interested on undertaking a query and later finish such as the telephone re-routing application in Figure 4 would be modelled as a *pull* consumer. On the other hand, a long-lived application such as the Jukebox Controller Application will constitute a push-style consumer waiting for its context abstractor's control notifications.

The Jukebox controller application in Figure 4 is the only TRIP-aware application implemented so far. This C++ application enables the control of a virtual software jukebox, implemented on top of an MPEG-3 player, through TRIP tags. Once initiated it registers with the *Jukebox Sighting Event Channel* as push consumer. Then according to the type of event it receives (see Listing 2), it executes the pertinent jukebox control action. For example, when the Jukebox Controller application receives a `MPEG3TrackEvent`, it initiates the playback of the song indicated in the event `songToPlay` attribute. Note this application was implemented as part of a bigger project regarding real-time multimedia streaming with CORBA, developed at AT&T Labs Cambridge.

5.2.4. The TRIP Directory Service Component

As it is observed in Figure 4, TRIP-aware context abstractors require of a database to map target IDs into entities. This requirement was already expressed in Section 3.1.1 when we identified the need for a centralised service that would regulate the TRIP code granting process, store static properties associated to TRIPcodes and provide interfaces for their query. The *TRIP Directory Service* CORBA component has been designed to answer these needs. This Python implemented component provides interfaces for the following operations:

1. Creation/Modification/Deletion of new categories of TRIPcodes.
2. Creation/Modification/Deletion of TRIPcodes and attributes associated to them.
3. Retrieval of a category's TRIPcodes and subcategories details.
4. Retrieval of a given TRIPcode's details.

Moreover, the TRIP Directory Server also provides an *asynchronous* notification mechanism for clients that only want to contact with it during their bootstrap stage and still be aware of modifications to their categories of interest. At its initialisation the service creates the *TRIP Directory Service Event Channel*, connects to it as a push supplier and binds its object reference with the Naming Service. Listing 3 shows the interface provided by the service to enable client application's to connect to its event channel. It also shows the IDL structure used to convey a TRIPcode creation notification. Similar events are generated to indicate modification and deletions of TRIPcodes and creations, modifications and deletions of categories of TRIPcodes. Appendix A provides a detailed explanation of the TRIP Directory Server implementation using UNIX *dbm* style files and the query interfaces exported to clients.

```
module TRIPDirectoryService {
  // Event interface:
  CosEventChannelAdmin::EventChannel getEventChannel();

  // Event structures:
  struct AddTRIPcodeEvent {
    string          categoryID;
    string          TRIPcode;
    TRIPcodeDetails details;
  };
(…)
};
```

Listing 3: TRIP Directory Service Event Interfaces in IDL

Figure 5 depicts how the TRIP Directory Service integrates into the SIS architecture. Interested parties such as the Jukebox Controller Context Abstractor obtain at their initialisation an object reference to the *TRIP Directory Service* that is used to invoke dictionary query operations as shown in Appendix A's Listing 4. Context abstractors then register to the *TRIP Directory Service Event Channel* to be notified when the persistent TRIP dictionaries are changed. For example, when a new TRIP code is created an event containing the structure shown in Listing 3 would be transmitted to the TRIP Directory Server Event Channel that would then forward it to all registered consumers. Note the combination of heterogeneous components written in different programming languages (C++ and Python) and over distinct CORBA ORBs (omniORB and Fnorb) that are interacting within the Sentient Information Service.

Potentially every item in the environment could be enforced to be attached a TRIP tag or at least to be assigned a TRIP code, given the wide range of possible TRIPcodes (1,594,323). For instance every camera feeding video frames to TRIP Monitoring Service Components could be identified by a TRIP code. This would enable the proposed TRIP Location Service Context Abstractor to map the camera ID provided by a TRIP sighting to its location by retrieving such data also from the TRIP Directory Service.

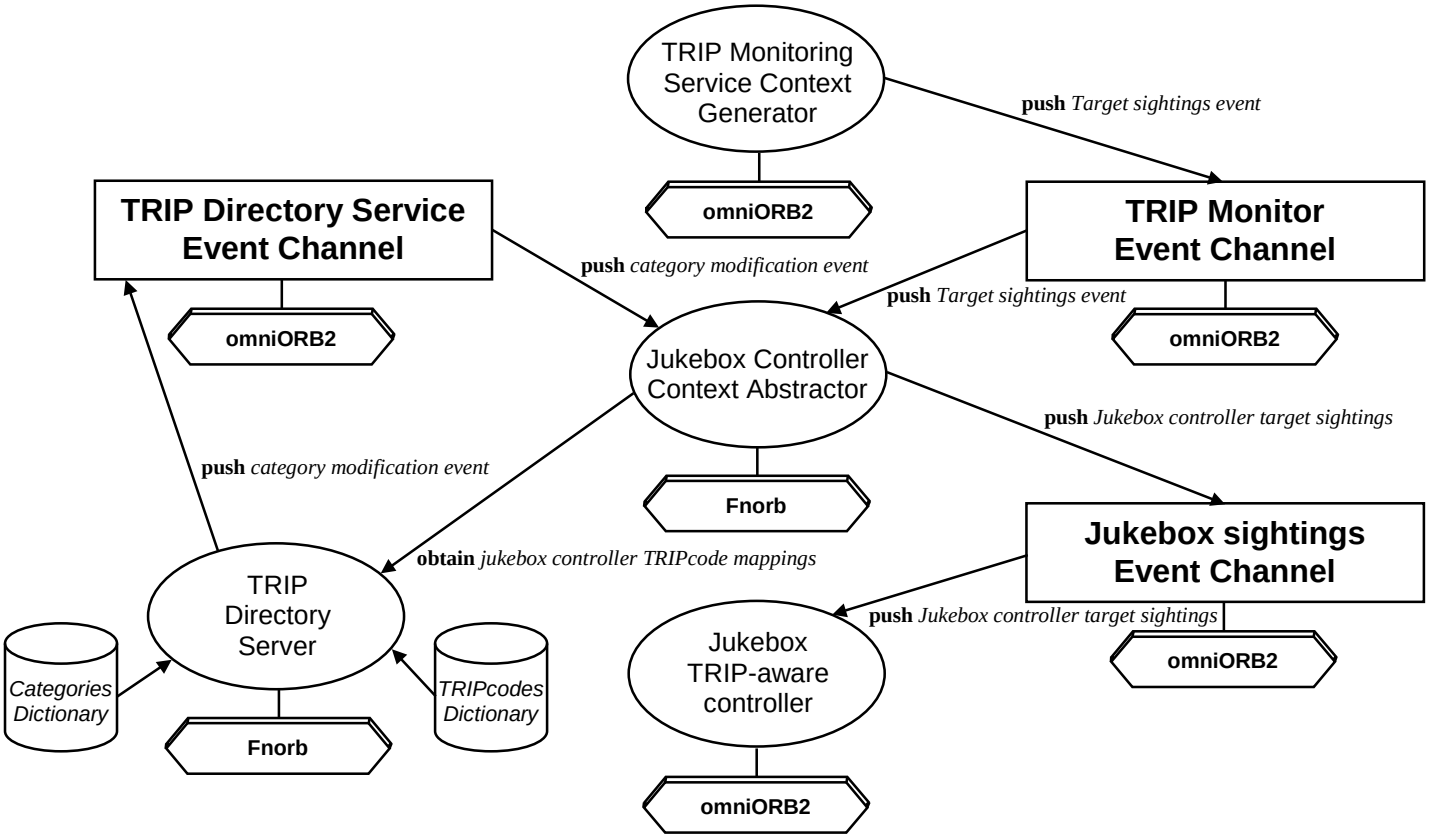

Figure 5: SIS TRIP-aware Components and Application

5.2.5. A GUI front-end for the TRIP Directory Server

In order to provide a user-friendly way to manage the creation, deletion, manipulation and query of TRIPcodes and categories a GUI front-end client for the TRIP Directory Server has been created. This client was implemented in Python using the multi-platform Pmw [Telstra99] GUI toolkit.

The TRIP Directory Server GUI-based client depicted in the upper part of Figure 6 is divided into two main interaction panes. The *TRIPcode Manager* Pane permits the user to: (1) browse through the existing TRIPcode categories, displaying their subcategories and TRIPcodes; (2) create/modify/delete subcategories; and (3) create TRIPcodes within a category. On the other hand, the *Search TRIPcode* pane provides the means to (1) query the information associated to a given TRIP target code and to (2) add, modify and delete its properties. The lower part of Figure 6 shows the result of double clicking over a TRIPcode list item. Note the *Print Target* facility provided by the TRIPcode visualisation dialog. A click over this button would send the POSTSCRIPT code generated for a TRIPtag to the printer. This capability facilitates highly the target generation process.

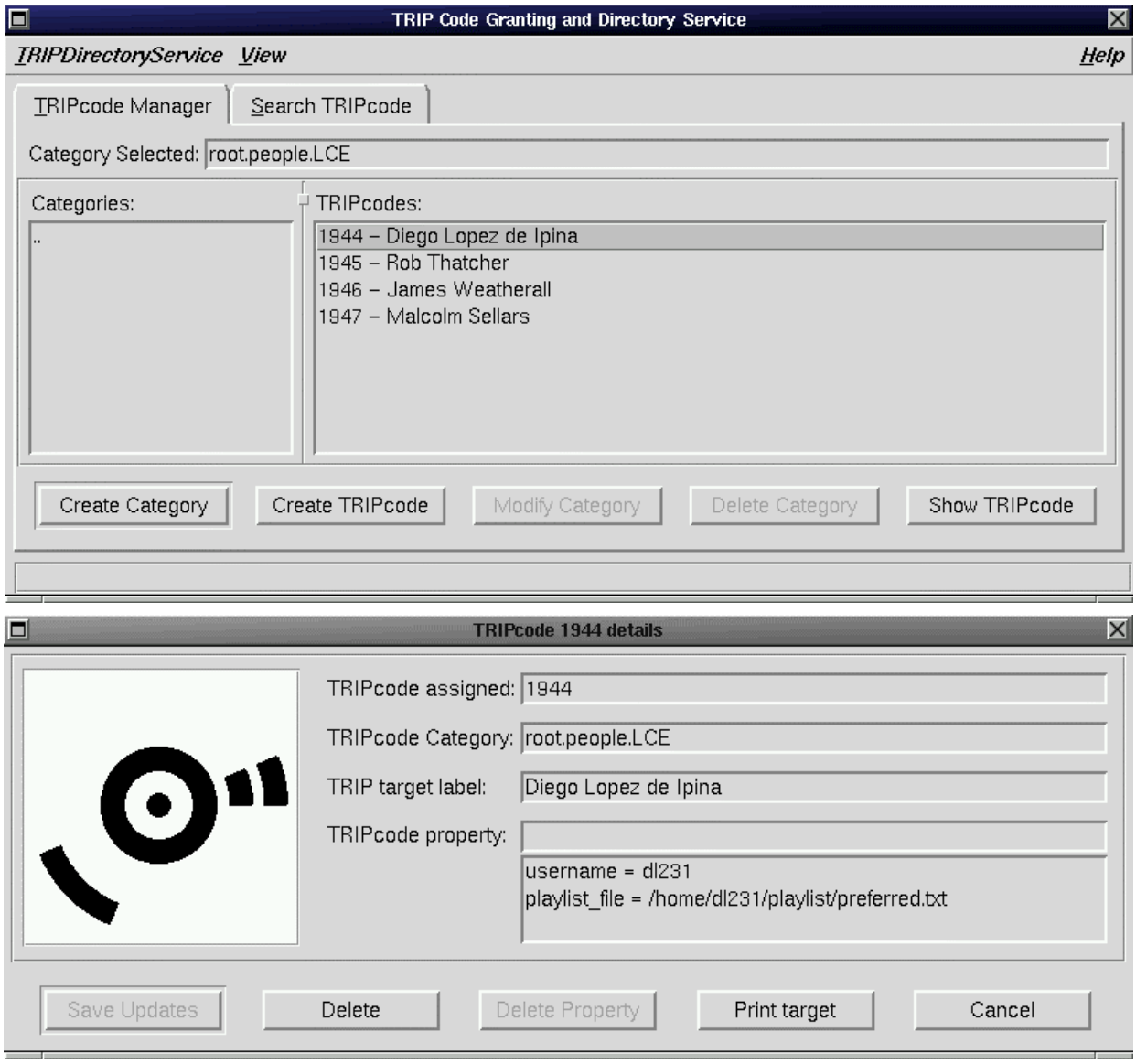

Figure 6: Snapshots of the TRIP Directory Client

6. The Sentient Information Service Architecture Evaluation

The SIS architecture described in Section 5 makes extensive use of the OMG Event Service's virtue to de-couple consumer and supplier objects by interposing intermediary Event Channel objects in between. Event Channels' adoption main benefit has been to enable asynchronous and transparent distribution of events and to provide mechanisms to mix consumer and suppliers in different event communication models. Suppliers were relieved from keeping registered consumer object references to perform a *distributed callback* every time an event had to be transmitted.

However, the OMG Event Service used presents some notorious deficiencies as it has been well described in [Smith97]. The first one is that the OMG Event Service specification [OMG98b] does not define requirements for some key Event Channel characteristics, and therefore the quality of service provided by Event Channels varies widely with the implementation. Some implementations employ unicast for event transmission whereas more efficient ones utilise multicast instead. Unfortunately the implementation we used, *omniEvents*, employs the first scheme. The second and most fundamental Event Service drawback is its *lack of filtering*. Event Channels pass events, in *any* IDL type, from their suppliers to their consumers without attempting to interpret event data in any way. Even if an Event Channel has only a single supplier connected to it, consumers like the Jukebox Context-Abstractor shown in Figure 5 may still receive events in which they are not interested. This situation leads to:

1. An increase in consumers' workload since they need to perform the filtering at the destination.
2. An increase in network utilisation due to the *dumb* delivery of all events to all consumers, even when no consumers may be interested in any of the events transmitted.

The architecture of the Sentient Information Service tries to workaround this lack of filtering of the Event Service by interleaving context abstractor components between the sources of contextual events (*TRIP Monitoring Service*) and the final targets of them (*TRIP-aware applications*). Context abstractors filter the events coming from context sources' Event Channels and guarantee final applications will receive relevant contextually enhanced events. Additionally, context-abstractors were co-located in the same address space of Event Channels to avoid unfiltered events network communication and employ inter-process communication instead. In order to minimise the unfiltered event traffic from Event Channels to context abstractors we also enforced the constraint that every Event Channel in our system had events generated by a single supplier type.

Fortunately, the OMG organisation has already adopted an enhanced version of the Event Service that attempts to overcome these problems. The OMG Notification Service [OMG98c] addresses the limitations of the Event Service, and supplies not only event filtering features but also various degrees of control over the quality of service that an Event Channel (here referred as *notification channel*) provides. In this new CORBA Service, consumers use predicate expressions to declare which events they are interested in receiving and convey them to the Notification Channels via subscription interfaces. When a supplier submits an event, the channel applies each

consumer's filter to determine whether that consumer is interested in receiving that event, reducing in this way unnecessary event notification traffic.

Future work in the SIS architecture will address the replacement of the Event Service by the Notification Service. This will ease context-abstractors programming significantly because their filtering task will be delegated to notification channels. Context abstractors will concentrate on aggregating and interpreting context to provide a high-level abstracted view of the environment to final applications. Simultaneously the event flow bandwidth consumption of the SIS will be reduced.

The current design of the Sentient Information Service is only concerned with providing context information regarding the current situation and lacks the desirable feature of a context persistence mechanism that would record past situations context. This would provide the means for potential context-based retrieval applications. In future work, this refinement of the architecture will be tackled.

Finally, it has to be admitted that thorough studies of the SIS throughput have not been produced yet to seriously assess the scalability of the architecture designed. Future work will address this study.

7. Future work

Work on TRIP will continue in three parallel areas. On one hand, research will be conducted to achieve a better processing rate and location resolution for the TRIP sensor technology. On other hand, the prototyped Sentient Information Service software architecture will be extended and probably partially re-designed to guarantee it is scalable enough to manage and distribute sentient data captured from many sensors, principally TRIP sensors, over a wide area and in real time. Finally, application development will be carried on to demonstrate the potential of TRIP cheap *Sentient Computing* technology. Hopefully, this last research effort will result in novel application areas not previously explored by already existing location technologies.

7.1. TRIP Sensor Technology Enhancements

7.1.1. Improving TRIP Sensor Recognition Rate

The TRIP sensor technology current status is able of processing 3 frames/second in a conventional PC machine. A main goal of this research is to improve this target recognition rate as much as possible. Video frames coming from several cameras should be processed simultaneously at enough speed to track entity movements through the environment. A 20-30 Hz performance rate to be achieved with a dedicated last generation PC is estimated as desirable. The following suggestions may help achieve this goal:

- *Parallelisation of the Target Recognition Process*.
  In the current implementation of the Target Recognition Process (remember Figure 2), the Adaptive Thresholding (stage 1) stage output is used by both edge detection (stage 2) and code recognition stages (stage 7). Stage 1 and 2 are by far the most CPU consuming stages as was mentioned in Section 4.2. While the Adaptive Thresholding stage is performed on one frame, the edge detection and the rest of the target recognition stages could be applied to another frame. This could be made feasible, running the Target Recognition Software on a machine with more than one CPU and making the recognition process multithreaded. Alternatively, several machines could be employed to make the target recognition processing faster, using parallel programming tools such as PVM [Geist97]. The performance of a target recognition pipeline could be reduced potentially to the time invested in the slowest stage plus the associated inter-machine communication costs.

- *Improve Target Recognition Algorithms.*
  Currently used computer vision algorithms will be revised and further optimised. A more thorough review in the literature will seek to identify alternative computer vision algorithms less computationally intensive.

7.1.2. Improving TRIP Sensor Location Resolution

TRIP currently provides *containment-based* location information, in the sense that can infer an entity location by indicating the camera view range space within which a TRIP tag is identified. The following suggestion pretends a better location resolution:

- *Apply Stereo Vision techniques to the TRIP technology*.
  A fine-grained location system giving 3-D positions of targets with respect to a plane of reference is aimed. Having images captured simultaneously from two different cameras, Stereo Vision techniques [Davies97] [Trucco98] can be applied to triangulate on features in both to infer depth. Providing the same TRIP target is spotted simultaneously by two video sources at fixed known locations, its 3-D position could be extracted.

  From a computational standpoint, a stereo system must solve two problems. The first known as *correspondence* problem, consists in determining which item in the left image corresponds to which item in the right one. TRIP solves this problem determining when the same target ID is identified in two different sources' frames. The second problem a stereo system must solve is *reconstruction*. This problem, given a number of corresponding parts on the left and right image, and information on the geometry of the stereo system, tries to determine the 3D location of the observed objects. Solving this reconstruction problem for TRIP will occupy the first priority in forthcoming work.

7.1.3. Target Redesign Considerations

Section 3.1.1 already mentioned the possibility of modifying the target design to provide a bigger range of identifiers or provide code encoding redundancy schemes that would enable the recognition of partially occluded TRIP tags. Future work will reconsider these considerations and evaluate alternative TRIP target designs.

7.2. Sentient Information Service Architecture Improvements

The SIS software architecture devised is a mere prototype and although it has helped us to experiment with the TRIP technology, it still has to suffer much improvement. Already some changes were proposed in Section 6. The following list summarises all the modifications intended:

- Adapt the SIS software architecture to replace the currently used OMG Event Service implementation (*omniEvents*) by an OMG Notification Service implementation, very probably DSTC's *CosNotification* [DSTC99].

- Create a *Location Service Context Abstractor* that: (1) receives TRIP sighting notifications from TRIP Monitoring Service components controlling TRIP sightings in each of the rooms of a building; (2) interprets such sightings and updates a cache with the last location of every entity tagged in the environment; (3) provides query interfaces to answer questions regarding the location of an entity and entities co-located in a place; and (4) notifies a *Location Service Event Channel* of all the entity position updates, where interested parties can connect.

- Generate a set of base reusable context-abstractors whose output can either be further abstracted by higher-level context abstractors or be used directly by final applications. For instance, a *RoomAbstractor* could be created that monitors people's TRIP target sightings within a room. Outputs from this abstractor and from an *Activity* context generator, reporting an event every time the sound level in a room goes higher than a threshold value, could be combined by a *MeetingAbstractor* to infer when the contextual conditions for a meeting are fulfilled.

- Consider new context abstractors with *sensor fusion* capabilities that merge location data coming from TRIP and other location technologies with similar capabilities such as Active Bat (e.g. *FineLocationAbstractor*) to generate canonical entity location events (e.g. *3DPositionEvent*).

- Prototype new context generators, preferably non-location related ones, such as the sound detector *Activity* context source previously mentioned to enable a more accurate picture of the environment.

- Produce a thorough analysis of the event dissemination capabilities of the Sentient Information Service. Study how the proposed *Location Service Context Abstractor* behaves in environments cluttered with a big number of TRIP tag wearers and in bursty situations when many users are moving simultaneously along a space. Examine different CORBA Notification Service implementations and their performance. If the event dissemination results obtained are not satisfactory enough, our own implementation of the standard CORBA Notification Service will be created, taking into account previous work on scalable Event Service architectures such as SIENA [Carzaniga98] or Cambridge Event Architecture [Bacon95] systems.

- Devise the hardware infrastructure required to create a reliable and cost-effective indoor location system based on TRIP technology. Determine the number of cameras necessary to provide full-coverage of a room and the number of TRIP processing servers. Deploy such system in a single room and compare its capabilities to already existing indoor location systems.

- Provide persistence capabilities to the SIS architecture to enable queries about past state of the environment. These historical queries will be based upon organised long-term storage of sensor data. Every context generator and abstractor may record all the context information they generate. For instance, the TRIP Monitoring Service could index captured video frames by TRIPcodes sighted, date and time. Applications will later be able to contact with SIS components to obtain past context information.

7.3. Applications in mind

TRIP-aware applications will be implemented as proof of concept of the different capabilities of the technology. The only exiting TRIP-aware application, the *Jukebox Controller*, has demonstrated the applicability of TRIP as an identification device. TRIP tags played the role of a user interface device that controls the operation of an

MPEG-3 player based virtual jukebox. New applications are intended to demonstrate TRIP suitability in the following areas:

- *TRIP as an Outdoors Identification Device*
  To demonstrate the suitability of TRIP as an identification device in outdoor environments a TRIP-aware *car park space detector* application is intended. Painting a target on each of the free car slots in a parking, the number of free available spaces could be sensed. A car user heading for her office could use the SMS service of her mobile phone to query an SMS Server [Stajano98] for the availability of parking space at the front and back of the office, thus avoiding a trial-and-error process that could be time consuming by morning traffic.

- *TRIP as a containment-based Location System.*
  TRIP technology current status could perfectly be used to deploy a containment-based indoor Location System. For that, the proposed Location Service Context Abstractor would have first to be implemented. Applications developed with existing containment-based indoor technologies such as the Active Badge could later be taken as model to create new ones making use of the TRIP Location Service.

- *TRIP as a fine-grain location technology.*
  Once the location resolution granularity of TRIP is enhanced from camera range of view resolution to 3D position, more sophisticated location-aware applications will be possible. A VR model of a room tracking TRIP tag wearers could for example be produced as Ward made previously with the Active Bat technology [Ward97]. Applications exploiting geometrical spatial relationships among objects in the environment as the Follow Me Bat Teleporting application described at [Harter99] could also be explored. To enable such applications a context abstractor should be developed with similar functionality to the Spatial Monitoring Service of the SPIRIT architecture [Steggles98].

- *TRIP as an Environment Monitoring Device.*
  Context captured by a TRIP-based indoor location system could be stored in a database. Questions such as ‘where a book was seen by last time’ or ‘who I was with last Monday in my office’ could be answered by a context-retrieval application using such database. Video recordings of office environments heavily tagged with TRIPcodes are also opened to the off-line processing and indexing of contextual information by TRIP target sighting. Applications based on the memory prosthesis system proposed by Lamming [Lamming94] will be attempted. These applications will be possible only after a persistent context mechanism is devised for the components of the SIS architecture.

- *TRIP as an enabling technology for Augmented Reality.*
  Previous work by Rekimoto [Rekimoto95] has employed a similar barcode technology to TRIP for Augmented Reality applications. TRIP target geometric features could be used as reference to register computer generated information on real world images. Hopefully, further improvements on TRIP location resolution will enable the author attempt similar applications.

7.4. Schedule of Work to be completed

- **OCTOBER 1999-MARCH 2000**
  - Apply Stereo Vision Techniques to improve location resolution.
  - Optimise Target Recognition's code to enhance TRIP performance.
  - Evaluate 3D location accuracy obtained with the created stereo system.
  - Devise the camera and processing infrastructure required for the creation of an indoor fine grain location system covering one room
  - Create a new *Location Service Context Abstractor* component that keeps track of the current location and orientation of each tagged object in the environment.
  - Implement an application demonstrating TRIP suitability as a fine grain indoor location technology and that makes use of the proposed Location Service Context Abstractor.

- **APRIL 2000-SEPTEMBER 2000**
  - Adapt the Sentient Information Service software architecture to accommodate the OMG Notification Service instead of the currently used Event Service.
  - Add support to the SIS architecture for the persistent storage of past context and historical query operations.
  - Implement a context-retrieval application undertaking historical queries.
  - Create a set of generic base context abstractors to ease development of applications and higher-level context abstractors.
  - Implement a *Spatial Monitoring Context Abstractor* that allows to determine geometric relationships between different tagged objects.
  - Evaluate the scalability of the architecture devised by undertaking simulations in a scenario where thousands of objects are spotted by multiple cameras distributed through a building.
  - Write the second year report

- **OCTOBER 2000-MARCH 2001**
  - Create new context generators to provide a more accurate model of the environment for sentient applications.
  - Create generic Location Abstractors merging location information provided by different tracking technologies that present similar capabilities.
  - Implement a *'killer'* application that shows the benefits of TRIP and the SIS architecture.
  - Integrate TRIP with the SPIRIT system at AT&T Laboratories Cambridge.
  - Devise some suitable metrics or decision factors to be used in comparing TRIP with similar existing location technologies.

- **APRIL 2001-SEPTEMBER 2001**
  - Write up Dissertation, collating references and results.

8. Conclusion

This report has described the work done by the author during his first year of PhD on the development of a vision-based sensor system that utilises easily printable small barcode tags attached to objects to identify them and infer their approximate location. A distributed architecture has also been built on top of this sensor system to enable the management and distribution of the sensorial data provided to applications. This architecture denominated *Sentient Information Service* has been employed to develop a first TRIP-aware application: the Jukebox Controller.

Further improvements on both TRIP technology and video cameras will permit obtaining accurate location of entities in the 3D space at a further distance from image sources. This accompanied by the enhancement of the current supportive distributed architecture, will permit the realisation of more sophisticated applications. It is hoped TRIP capability to tag huge number of entities will be especially useful in the context-retrieval applications domain. A schedule of the work to be done has shown the research activities planned for the next two years of PhD.

Acknowledgements

I would like to express my gratitude to Sai-Lai Lo and Joe Newman at AT&T Laboratories Cambridge for their assistance and advice during this work. Sai-Lai provided the frame grabber software employed in this project and assisted greatly in the first TRIP-aware application development. Joe undertook some preliminary work on TRIP that was of great help for the author. Thanks as well for Prof. Andy Hopper for suggesting and giving me the opportunity to research in this fascinating TRIP project. Finally I would like to acknowledge AT&T for the industrial sponsorship of this project, and to the Basque Government for its financial support to my PhD studies.

Appendix A. TRIP Directory Service Implementation

The Python [van Rossum99a] scripting language was used in the TRIP Directory Service Implementation. This language was chosen because it enables very rapid development of applications and still is general enough to provide the programming features we required for the TRIP Directory Service component: CORBA-support and key-based object persistence. Fnorb [Chilvers99] Python CORBA mapping implementation and Python's standard library *shelve* (inspired on the UNIX *dbm* files) key-based object persistence module provided such features.

CORBA clear separation of implementation and interfaces by means of IDL makes possible to re-implement a component in a more efficient programming language later without having to modify the code of its clients. In our case, speed of development resulted determinant to choose Python albeit the obvious sacrifice of execution efficiency. If scalability or performance problems appear in the future, always the TRIP Directory Server could be re-coded in a more performance critical system programming language (C++ or Java) and/or a commercial DBMS engine or X.500 Directory Service could be used instead of the basic *shelve* persistence mechanism.

A.1. TRIP Directory Server Persistent Dictionaries

The TRIP Directory Service persistence mechanism is implemented through two persistent associative arrays or *shelves*:

- The *Categories Shelve* contains category nodes hashed by *categoryKey*. A categoryKey is a string with the format $(xxx)^+$, where *xxx* is a 3 digits ternary code in the range [000-212][7] and '+' denotes 1 or more of these sequences. Each category node is, at the same time, a dictionary by itself. Figure 7 shows a node structure in this dictionary.

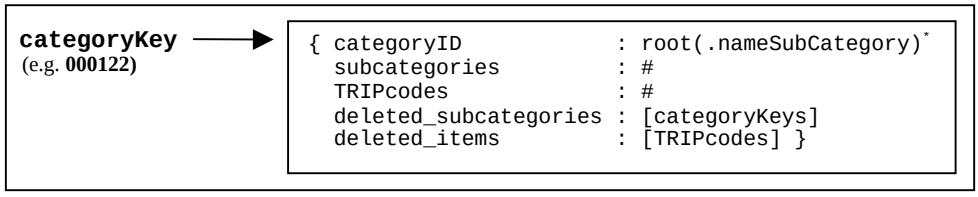

Figure 7: Categories Dictionary Node

- The *TRIPcodes Shelve* associates TRIPcodes, the ternary representation of a number in the range [0-1594322[8]], to a Python mapped CORBA IDL structure (see Figure 8) containing as members a sequence of name/value pairs and a label.

For each category up to 24 subcategories can be created (range [000-212]). When a new subcategory is assigned, the identifier of the new category is formed by adding to the parent's *categoryKey* the following non-used ternary code string in the range [000-212]. A TRIPcode is composed of a prefix with the key of its category, followed

[7] Note the range [220-222] is reserved to denote the beginning of a valid TRIPcode.
[8] $1594322 = 3^{13} - 1$, is the value of the maximum identifier number that can be assigned to a TRIPcode.

by the ternary string *'22'*, and the remaining ternary digits up to the 13 the design of TRIP targets address supports, with the target sequence number within its category.

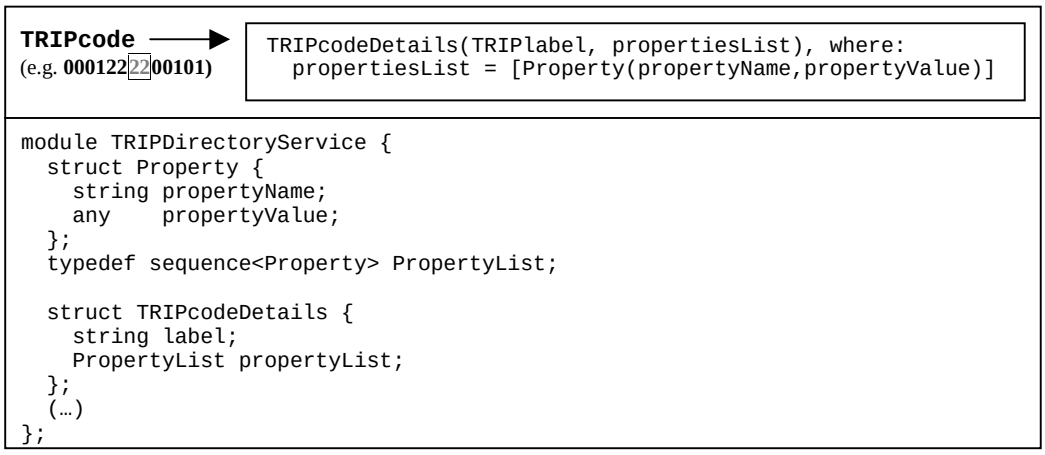

Figure 8: TRIPcodes Dictionary Node and its associated IDL structures

A.2. TRIP Directory Server Functionality

The following IDL code snippet represents the interfaces offered by the TRIP Directory Server to its clients:

```
module TRIPDirectoryService {
  (…)
  interface TRIPDirectoryServiceIF {
    # TRIPcode Dictionary manipulation interfaces
    string grantTRIPCode(in string categoryID);
    void   saveTRIPcode(in string TRIPcode, in TRIPcodeDetails data);
    void   saveTRIPcodeUpdates(in string TRIPcode,in TRIPcodeDetails data);
    void   deleteTRIPcode(in string TRIPcode);

    # Categories Dictionary manipulation interfaces
    boolean createCategory(in string parentCategoryID, in string categoryName);
    void    updateCategoryName(in string oldCategoryID, in string newCategoryID);
    void    deleteCategory(in string categoryID);

    # Query interfaces for Categories Dictionary
    stringList getSubCategoriesList(in string categoryID);
    stringList getCategoryTRIPcodes(in string categoryID);
    string     getCategoryKey(in string categoryID);

    # Query interfaces for TRIPcodes Dictionary
    TRIPcodeDetails getTRIPcodeDetails(in string TRIPcode);
    (…)
  };
};
```

Listing 4: TRIP Directory Server interfaces in IDL

References

[Adly97] Adly N., Steggles P. and Harter A. “SPIRIT: a Resource Database for Mobile Users”, Proceedings of ACM CHI'97 Workshop on Ubiquitous Computing, Atlanta, Georgia, March 1997

[ANSA90] “Advanced Networked Systems Architecture”, Architecture Projects Management Limited, Poseidon House, Castle Park, UK, August 1990.

[Azuma97] Azuma R. “A Survey of Augmented Reality”. Presence: Teleoperators and Virtual Environments 6, 4, (August 1997), 355-385

[Azuma99] Azuma R. “The Challenge of Making Augmented Reality Work Outdoors”. “Mixed Reality: Merging Real and Virtual Worlds” by Yuichi Ohta and Hideyuki Tamura, Springer-Verlag, Chp21 pp. 379-390. ISBN 3-540-65623-5, 1999

[Bacon95] Bacon J, Bates J., Hayton R. and Moody K. “Using Events to Build Distributed Applications”. Proceedings IEEE Services in Distributed and Networked Environments, pp. 148-155, Whistler, June 1995

[BARCODE98] “Information about Barcodes”, The Barcode Software Center, 1998, http://www.mecsw.com/info.html

[Bates96] Bates J, Halls D, and Bacon J. “A Framework to Support Mobile Users of Multimedia Applications”, ACM Mobile Networks and Nomadic Applications (NOMAD), 1(4), pp 409-419, 1996.

[Beadle98] Beadle H., Maguire G., Smith M. “Location Based Personal Mobile Computing and Communication”. Proceedings 9th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN'98), Banff, Alberta, Canada, 17-20 May 1998, pp. 23-24.

[Brown96] Brown P. “The stick-e document: a framework for creating context-aware applications”. In Proceedings of EP'96, Palo Alto, pp. 259-272, January 1996.

[Brown97] Brown P.J., Bovey J. D. and Chen X. “Context-aware applications: from the laboratory to the marketplace”, IEEE Personal Communications 4(5), 58-64, 1997

[Carzaniga98] Carzaniga A., Rosenblum D. and Wolf A. “Design of a Scalable Event Notification Service: Interface and Architecture”. Technical Report CU-CS-863-98, Department of Computer Science, University of Colorado, August 1998

[Chilvers99] Chilvers, M., ”Fnorb – Version 1.0”, Distributed Systems Technology Centre, University of Queensland, Brisbane, Australia, April 1999, http://www.dstc.edu.au/Products/Fnorb/user-guide.html

[Dana98] Dana P. “Global Positioning System Overview”, Department of Geography, University of Texas at Austin, 1998 http://www.utexas.edu/depts/grg/gcraft/notes/gps/gps.html

[Davies97] Davies E. “Machine Vision – Theory, Algorithms, Practicalities” The Three Dimensional World, Chapter 15, pp. 373-417, Academic Press, 1997 ISBN 0-12-206092-X

[Dey99a] 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 (UIST '99), 1999.

[Dey99b] Dey A., Futakawa M, Salber D. and Abowd G.. “The Conference Assistant: Combining Context-Awareness with Wearable Computing”. To appear in the Proceedings of the 3rd International Symposium on Wearable Computers (ISWC '99), San Francisco, CA, October 20-21, 1999.

[DSTC99] “CosNotification: An OMG CORBA Notification Service Implementation”, Distributed Systems Technology Centre, University of Queensland, Brisbane, Australia http://www.dstc.edu.au/Products/CORBA/Notification_Service/

[Fitzgibbon96] Fitzgibbon A., Pilu M. and Fisher R. “Direct least squares fitting of ellipses”. Proceedings of International Conference on Pattern Recognition, August 1996.

[Geist97] Geist A. “Advanced Tutorial on PVM 3.4 New Features and Capabilities”,, CSM Oak Ridge National Laboratory, 1997 http://www.epm.ornl.gov/pvm/EuroPVM97/

[Harter94] Harter A., Hopper A. “A Distributed Location System for the Active Office”, IEEE Network, Vol. 8, No. 1, January 1994

[Harter99] 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

[Hull97] Hull R., Neaves P. and Bedford-Roberts J. “Towards situated computing”. Proceedings of International Symposium on Wearable Computers, Boston, IEEE October 1997, pp. 146-153

[Kirsch97] Kirsch D. and Starner T. “The Locust Swarm: An environmentally-powered, networkless location and messaging system”, Proceedings of the 1st International Symposium on Wearable Computers, pp. 169170, October 1997

[Lamming94] Lamming M. and Flynn M."Forget-me-not" Intimate Computing in Support of Human Memory, Proceedings of FRIEND21, ’94 International Symposium on Next Generation Human Interface, Japan, February 1994

[Lo99] Lo S, Riddoch D, “The omniORB2 version 2.7.1 User's Guide ”, AT&T Labs Cambridge, UK, February 1999, http://www.uk.research.att.com/omniORB/doc/omniORB2/omniORB2.html

[Nelson98] Nelson G. “Context-Aware and Location Systems”. PhD Thesis. Cambridge University Computer Lab, UK, January 1998

[OMG98a] OMG, Object Management Group, “CORBA/IIOP 2.2 Specification”, February 1998, ftp://ftp.omg.org/pub/docs/formal/98-07-01.pdf

[OMG98b] OMG, Object Management Group, “CORBA Services: Common Object Services Specification”, September 1998, ftp://ftp.omg.org/pub/docs/formal/98-12-09.pdf

[OMG98c] OMG, Object Management Group, “Notification Service – Joint Revised Submission”, November 1998, ftp://ftp.omg.org/pub/docs/telecom/98-11-01.pdf

[Pascoe98] Pascoe J. “Adding Generic Contextual Capabilities to Wearable Computers”, Proceedings of the 2ND. International Symposium on Wearable Computers, pp. 92-99, 1998

[PINPOINT98] “3D-iD Technology and Features”, 1998
http://www.pinpointco.com/technology/technology.htm

[Rekimoto95] Rekimoto J. and Nagao K. “The World through the Computer: Computer Augmented Interaction with Real World Environments”, User Interface Software and Technology (UIST’95), 1995

[Rekimoto98] Rekimoto J. “Matrix: A Realtime Object Identification and Registration Method for Augmented Reality”. Proceedings of the Asia Pacific Computer Human Interaction Conference (APCHI’98), Japan, July 1998

[Richardson94] “Teleporting in an X Window System Environment”. IEEE Personal Communications Magazine, Vol. 1, No. 3, Third Quarter 1994, pp 6-12.

[Richardson98] Richardson T., Stafford-Fraser Q., Wood K. and Hopper A. “Virtual Network Computing”. IEEE Internet Computing, Vol. 2, No. 1, 1998, pp. 33-38

[Schilit94a] Schilit B., Adams N., and Want R. “Context-Aware Computing Applications “, Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CAIEEE Computer Society, December 1994.

[Schilit94b] Schilit B. and Theimer M. “Disseminating active map information to mobile hosts”. IEEE Network, 1994

[Sedgewick94] Sedgewick R. “Algorithms” Prentice-Hall Inc., 2nd Edition, Page 353, 1994, ISBN: 0-201-06673-4

[Smith97] Smith D., Vinoski S. “Overcoming Drawbacks in the OMG Event Service”, SIGS C++ Report magazine, June 1997, http://www.iona.com/hyplan/vinoski/#columns,

[Stajano98] F.Stajano and A.Jones, “The Thinnest Of Clients: Controlling It All Via Cellphone”. ACM Mobile Computing and Communications Review, vol 2 no 4, October 1998

[Starner97] Starner T., Mann S., Rhodes B. et al “Augmented Reality Through Wearable Computing”, Presence, Special Issue on Augmented Reality, 1997

[Steggles98] Steggles P., Webster P., Harter A. “The Implementation of a Distributed Framework to support 'Follow Me' Applications”, AT&T Laboratories Cambridge, Technical Report 98.8, 1998

[Szymaszek98] Szymaszek J., Uszok A. and Zielinski K. “Building a Scalable and Efficient Component Oriented System using CORBA – an Active Badge System Case Study”, Proceedings 4th Conference on Object-Oriented Technologies and Systems (COOTS'98), April 1998, Santa Fe, New Mexico.

[Tanenbaum96] Tanenbaum A. ”Computer Networks”, 3rd Edition, Prentice Hall, 1996, ISBN 0-13-349945-6

[Telstra99] Telstra Corporation Limited, Australia , “Pmw Python megawidgets”, June 1999, http://www.dscpl.com.au/pmw/

[TIRIS98] The Texas Instruments Registration and Identification System Home Page http://www.ti.com/mc/docs/tiris/index.html

[Trucco98] Trucco E. and Verri A. “Introductory techniques for 3-D Computer Vision”, Prentice-Hall, Inc. 1998, ISBN-0-13-261108-2

[van Rossum99a] van Rossum G. “Python Tutorial - Release 1.5.2”. Corporation for National Research Initiatives (CNRI), , July 6, 1999, http://www.python.org/doc/current/tut/tut.html

[Want92] 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

[Want95] Want R., Schilit B., Adams N., Gold R., Goldberg D., Petersen K., Ellis J., Weiser M., "An Overview of the Parctab Ubiquitous Computing Experiment", IEEE Personal Communications, Vol 2. No.6, pp28-43, December 1995

[Ward97] Ward A., Jones A. and Hopper A. “A New Location Technique for the Active Office”, IEEE Personal Communications, October 1997, pp. 42-47

[Ward98] Ward A. “Sensor-driven Computing”. PhD Thesis. Cambridge University Engineering Department, UK, August 1998

[Weiser93] Weiser M. “Some Computer Science Issues in Ubiquitous Computing”. Communications of. ACM 36, 7 (Jul. 1993), Pages 75 – 84

[Wellner93] Wellner P. “Interacting with paper on the DigitalDesk”. Communications of the ACM, 36(7): 87-96, Aug. 1993.

[Wren97] Wren C., Azarbayejani A., Darrell T. and Pentland A. “Pfinder: Real-Time tracking of the Human Body”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No. 7, July 1997