Technical Report

UCAM-CL-TR-686
ISSN 1476-2986

Number 686

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Dependable systems for Sentient Computing

Andrew C. Rice

May 2007

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Abstract

Dependable Systems for Sentient Computing
Andrew Rice

Computers and electronic devices are continuing to proliferate throughout our lives. Sentient Computing systems aim to reduce the time and effort required to interact with these devices by composing them into systems which fade into the background of the user's perception. Failures are a significant problem in this scenario because their occurrence will pull the system into the foreground as the user attempts to discover and understand the fault. However, attempting to exist and interact with users in a real, unpredictable, physical environment rather than a well-constrained virtual environment makes failures inevitable.

This dissertation describes a study of *dependability*. A dependable system permits applications to discover the extent of failures and to adapt accordingly such that their continued behaviour is intuitive to users of the system.

Cantag, a reliable marker-based machine-vision system, has been developed to aid the investigation of dependability. The description of Cantag includes specific contributions for marker tracking such as rotationally invariant coding schemes and reliable back-projection for circular tags. An analysis of Cantag's theoretical performance is presented and compared to its real-world behaviour. This analysis is used to develop optimised tag designs and performance metrics. The use of validation is proposed to permit runtime calculation of observable metrics and verification of system components. Formal proof methods are combined with a logical validation framework to show the validity of performance optimisations.

Contents

List of Figures **9**

Acknowledgements **13**

Publications **14**

1 Introduction **15**

1.1 Dependable Computing 17

1.1.1 Existing technologies 17

1.1.2 Engineering for dependability 17

1.1.3 Algorithmic dependability 18

1.1.4 Runtime dependability 18

2 Related Work **19**

2.1 Supporting Sentient Applications 19

2.1.1 Operating environment 19

2.1.2 Coverage and boundaries 20

2.1.3 Types of location information 21

2.1.4 Information delivery 22

2.1.5 Direction of observation 23

2.1.6 Search constraints 23

2.1.7 Statefulness 24

2.1.8 Sensing interface 25

2.1.9 Calibration 35

2.2 Uncertainty in Location Information 36

2.3 Fault Tolerant Systems 38

2.3.1 Hardware failure 39

2.3.2 Redundancy . . . 40
2.3.3 Fault tree analysis . . . 41
2.3.4 State dependent analysis . . . 41
2.3.5 Software . . . 42
2.4 Summary . . . 43

3 A Platform For Investigating Dependability **45**
3.1 Design Goals . . . 45
3.2 Machine Vision Systems . . . 46
3.3 Fundamentals of Tag Design . . . 47
3.3.1 Data coding . . . 47
3.3.2 Tag shape . . . 49
3.4 Image Processing Pipeline . . . 52
3.4.1 Entity abstraction . . . 52
3.4.2 Image acquisition . . . 54
3.4.3 Thresholding . . . 54
3.4.4 Contour following . . . 58
3.4.5 Camera distortion correction . . . 58
3.4.6 Shape fitting . . . 59
3.4.7 Transformation . . . 60
3.5 Dependable Coding . . . 62
3.5.1 Rotational invariance . . . 63
3.5.2 Tag coding abstraction . . . 65
3.5.3 Coding schemes . . . 66
3.5.4 Asymmetric tags . . . 68
3.5.5 Evaluation . . . 69
3.6 Back-Projection for Circular Tags . . . 71
3.6.1 Algorithm description . . . 71
3.6.2 Evaluation . . . 75
3.6.3 Resolving pose ambiguity . . . 75
3.7 Summary . . . 77

4 Algorithmic Dependability **79**
4.1 Specifying Performance . . . 79
4.2 Features of the Camera Model . . . 80
4.3 Sample Distance . . . 82
4.3.1 Circular tag design . . . 84
4.3.2 Comparing square and circular tags . . . 90
4.3.3 Error-correcting coding schemes . . . 91
4.3.4 Implications for system deployment . . . 92
4.4 Sample Error . . . 93
4.5 Sample Strength . . . 94
4.5.1 Estimated sample strength . . . 95
4.5.2 Real-world tag reading performance . . . 97
4.5.3 Summary . . . 99
4.6 Location Accuracy . . . 101
4.7 Achieving Algorithmic Dependability . . . 103
4.8 Summary . . . 104

5 Runtime Dependability **107**
5.1 Runtime Faults in Sentient Computing . . . 107
5.2 Exploiting Asymmetric Computation . . . 108
5.2.1 Negative validation . . . 109
5.3 Expressing Validation Requirements . . . 109
5.4 A Validation Architecture for Cantag . . . 111
5.5 A Validation Reasoning Engine . . . 112
5.5.1 Implementing the check predicates . . . 115
5.5.2 The validation process and costs . . . 115
5.5.3 Improving performance . . . 116
5.6 Application-Oriented Validation . . . 121
5.6.1 The LAST predicate . . . 123
5.6.2 Entry events . . . 123
5.7 Validation for the Active Bat system . . . 125
5.7.1 Improving performance for the LAST predicate . . . 127
5.7.2 Ensuring system safety . . . 129
5.8 Usage Modes . . . 131
5.9 Implementation Considerations . . . 132
5.9.1 Validating historical events . . . 133
5.10 Summary . . . 134

6 Conclusion **135**

6.1 Investigation Platform 135

6.2 Performance Metrics 136

6.3 Validation 136

6.4 Future Work 137

6.5 Summary 137

References **139**

Figures

1.1 Percentage of households with electronic devices in the UK 15

2.1 The SPIRIT map application showing *computer-visible* regions 22
2.2 Position error in the Active Bat system due to multipath signals 29
2.3 Relative spectral emissions from selected light sources 30
2.4 Accuracy and precision of location estimates 36
2.5 Location expressed as a Probability Density Function 37
2.6 The *bathtub* curve for the probability of hardware failure 39

3.1 MBV Systems using square tag designs 48
3.2 MBV Systems using circular tag designs 48
3.3 Ambiguous interpretations for the pose of a circular tag 51
3.4 Tag design terminology 51
3.5 The three generalised circular tag designs provided in Cantag 51
3.6 A simple implementation of type-lists in C++ 53
3.7 The ComposedEntity class 53
3.8 Inheritance diagram for an example ComposedEntity 54
3.9 Noise sources in images captured from CCD arrays 55
3.10 Photon shot noise in equalised low-illumination images 56
3.11 Optical illusion in colour interpretation 56
3.12 Thresholding under varying lighting conditions 57
3.13 Thresholding images taken in low illumination conditions 57
3.14 Perspective effects on the TransformEllipseLinear algorithm 61
3.15 Large payloads are susceptible to perspective errors 61
3.16 Reading a circular tag non-invariantly or invariantly 64
3.17 Reading a square tag in a rotationally invariant manner 64
3.18 The central datacell is unused if the tag has an odd number of cells 65

3.19 The Independent Chunk Code applied to a tag with large symbols 67
3.20 Data-carrying capabilities of the evaluated coding schemes 69
3.21 Error rates for the evaluated circular tag designs 70
3.22 Error rates for the evaluated square tag designs 71
3.23 Comparing TransformEllipseLinear to TransformEllipseFull 75
3.24 Tag decoding performance of square and circular tags 76

4.1 The error in the tag's normal vector is bimodal 80
4.2 Tag size is inversely proportional to distance from the camera 81
4.3 Full tag sizes for a number of example camera configurations 81
4.4 Tag size with fixed orientation for positions along a ray 82
4.5 Sample distances for three example tags 83
4.6 Analogy to the Nyquist-Shannon sampling limit 83
4.7 Pixel aliasing affect on minimum sample distance 84
4.8 Approximating the original TRIP tag design 84
4.9 TRIP tag interpolated tag size for increasing payload size 85
4.10 Tangential and radial size calculation 86
4.11 Interpolated tag size of CircleInner tags against payload size 89
4.12 Comparing CircleInner tags with the remaining circular designs 89
4.13 Comparing CircleInner and Square tags 90
4.14 The effect of position and pose on interpolated tag size 91
4.15 The number of unreadable datacells for selected tag inclinations 92
4.16 The required tag size, per datacell, for one pixel of sample distance 93
4.17 Sample error between the ideal and estimated sample points 93
4.18 Maximum sample error from FitEllipseLS and FitEllipseSimple 94
4.19 Sample strength of FitEllipseLS and FitEllipseSimple 95
4.20 Approximation Error when estimating Sample Strength 96
4.21 Cumulative Error for the Estimated Sample Strength 98
4.22 Experimental Setup for real-world testing of Cantag 99
4.23 Sample Strength Estimates using real-world data 100
4.24 Real-world and simulated location error across processing pipelines 102

5.1 Functional diagram of a Cantag processing pipeline 111
5.2 Inference rules for validation in Cantag 113

5.3 Validation rules for Cantag (Prolog clauses) 114
5.4 System validation (Prolog clauses) 115
5.5 An example validation session for Cantag 116
5.6 The number of calls to positive and negative checking functions 117
5.7 Classification of the computation costs of the checking function 118
5.8 Early rejection of candidate entities reduces validation costs 120
5.9 Negative validation rules for the CONT selection predicate 122
5.10 Inference rules for LAST_p 123
5.11 Validation of CONT and EXCL (Prolog clauses) 124
5.12 Implementation of LAST (Prolog clauses) 124
5.13 Functional diagram of the Active Bat system 125
5.14 Inference rules for validation in the Active Bat System 126
5.15 Validation inference rules for the ID predicate 127
5.16 Specification of validity proof for $(\mathrm{LAST}*)$ and $(\mathrm{LAST}*_2)$ 129
5.17 Proof of soundness for the $\mathrm{LAST}*$ predicates (Isabelle/HOL) 130
5.18 Proof of validity for the implementation of $\mathcal{S}_{\mathrm{ID}}$ (Isabelle/HOL) 131

Acknowledgements

Many thanks are due to my supervisor Andy Hopper for his direction, insights, and guidance. I am grateful to Alan Mycroft for his acute observations and suggestions particularly in regard to C++ programming techniques, the information-theoretic limits of Cantag, and theorem proving. Alastair Beresford and Robert Harle have been of great help by providing implementations for various parts of Cantag such as the eigenvector solving routine, and algorithms for back-projection of square tags. Rotational invariance for tag coding arose from discussions with Christopher Cain. I am further indebted to him for the development and implementation of the Structured Cyclic Code (SCC) coding scheme. Further thanks go to my colleagues David Cottingham, Jon Davies, John Fawcett, Brian Jones, Matthew Parkinson, Tom Ridge, Richard Sharp, Eben Upton, and Ian Wassell for their academic guidance and proof-reading. Personal thanks go to Ewan Mellor and Cheryl O'Rourke for their support; to Paula Buttery for her love and companionship; and to my parents Victor and Lindsay.

This thesis was examined by Professor Gudrun Klinker and Professor Peter Robinson. I thank them for their time and their comments.

I gratefully acknowledge the financial support of the EPSRC.

Publications

A number of the contributions presented in this work have appeared in the following publications:

- Andrew Rice, Christopher Cain and John Fawcett. Dependable Coding for Fiducial Tags. In *Proceedings of the 2nd Ubiquitous Computing Symposium*, pages 155–163, 2004.
- Andrew Rice, Christopher Cain and John Fawcett. Dependable Coding for Fiducial Tags (Extended Version). In *Ubiquitous Computing Systems, LNCS 3598*, pages 259–274, 2004.
- Andrew Rice and Robert Harle. Evaluating Lateration-Based Positioning Algorithms for Fine-Grained Tracking. In *Joint Workshop on Foundations of Mobile Computing (DIAL-M-POMC)*, pages 54–61, 2005.
- Andrew C Rice, Alastair R Beresford and Robert K Harle. Cantag: an open source software toolkit for designing and deploying marker-based vision systems. In *Fourth Annual IEEE International Conference on Pervasive Computer and Communications (PerCom)*, pages 12–21, 2006.
- Andrew C Rice and Alastair R Beresford. Dependability and Accountability for Context-aware Middleware Systems. In *Workshop on Middleware Support for Pervasive Computing Workshop (PerWare)*, pages 378–382, 2006.
- Andrew C Rice, Robert K Harle and Alastair R Beresford. Analysing fundamental properties of marker-based vision system designs. In *Pervasive and Mobile Computing*, 2(4):453–471, 2006.

Chapter 1

Introduction

Computers and electronic devices are continuing to proliferate throughout our lives. The time and effort required to control or interact with these devices is increasing with their number and heterogeneity.

Our surroundings are burgeoning with electronic devices such as television sets, Personal Video Recorders (PVRs),control systems for central heating, microwave ovens, and in-car satellite-assisted navigation units. The number of mobile devices is also increasing. Examples of these include mobile telephones, digital cameras, and digital music players. Figure 1.1 shows evidence of these trends at the national level in the United Kingdom [52].

Communication between devices is becoming commonplace: radio and infrared networking capabilities are ubiquitous in mobile telephones; digital cameras can connect directly to personal computers; and televisions (and refrigerators) are now connected to the Internet. Acquisition of a new device combinatorially increases the complexity of our situation because it may potentially interact with all our existing devices. This is analogous to the problem of adding developers late in a software engineering project—many new lines of communication are created for each addition [24, Chapter 2].

The ultimate goal of this work is to reduce the strain on users by **minimising the cognitive load of using electronic devices**. This complements the vision of Ubiquitous Computing [148] which aims to create technologies that fade into the background.

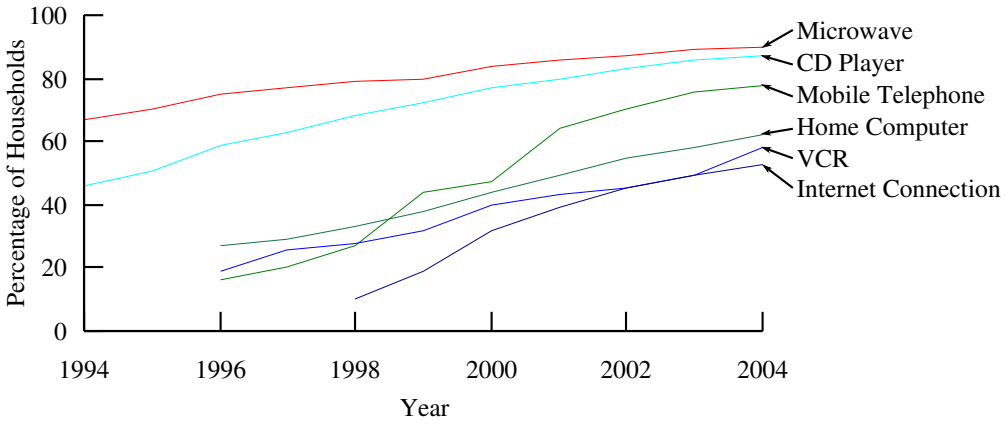


Figure 1.1: Percentage of households with electronic devices in the UK

Research into Sentient Computing seeks to achieve this goal by shifting the onus of understanding from user to machine [75]. Machines which understand their users should be easier to use than current devices and might ultimately require no direct input at all. Currently, applications operate in a virtual environment inside the computer and we interact with them using abstract control devices such as keyboards and mice. Sentient Computing seeks to move this virtual, abstract interface into our physical, real-world environment. This is broadly in-line with the goal of Proactive Computing [135] which aims for computer systems to: 1) *get physical* through the use of sensors to achieve coupling with the environment; 2) *get real* by safely and rapidly responding to external stimuli; and 3) *get out* by moving human operators above the operating loop.

One technique for interacting with users in the physical environment is to make applications and devices *context-aware*. Numerous forms of context exist that an application might utilise. Four of the most important of these are location, identity, activity, and time [40].

Substantial progress has been made towards context-aware interaction. Location information has received particular attention. At the lowest level, myriad systems have been developed for collecting location information with differing degrees of accuracy and precision. This information has been used by applications directly for location cues, and also to infer other forms of context such as identity [18] and activity [67]. Research into middleware, which run on top of location systems, has developed concepts such as spatial call-backs which assist applications running on low resource platforms by providing asynchronous notification when an event of interest occurs [2]. At the top level, many applications have been developed which exploit this information.

Sentient Computing systems are extremely *failure sensitive*. From a conceptual standpoint it is impossible for the system to become invisible its users observe an appreciable rate of failures occurring. Due to its pervasive nature, any attempted deployment of Sentient Computing is likely to be thwarted by failures because users will not trust a fragile system.

In addition to being failure sensitive, Sentient Computing systems are also *failure prone*. This is because, by definition, these systems are attempting to exist and interact with users in a real, unpredictable, physical environment rather than a well-constrained virtual environment.

The conventional engineering approach to improving systems of this nature is to reduce the failure rate using fault tolerant hardware and software techniques. These techniques are also applicable within Sentient Computing systems but their application must not detract from other aspects of the system: common goals for devices and systems such as miniaturisation or mobility impose size and power consumption constraints which limit the efficacy of fault tolerant engineering.

A dependable system can provide, at any time, a specification of current system performance and status. Applications are able to determine when a failure occurs and adapt accordingly. The system's dependability is the proportion of time that the actual service level matches the advertised service level. Dependability aims to reduce the failure sensitivity of Sentient Computing by enabling applications to adapt to faults and continue to operate to the best possible extent.

1.1 Dependable Computing

The contribution embodied in this work is a structured approach to implementing dependable systems for Sentient Computing. Specifically, this consists of:

- implementation of a location system with the specific goal of supporting dependability;
- particular improvements to the robustness of machine vision algorithms used in location systems;
- a process for analysing the performance of a location system and deriving performance metrics;
- integration of these metrics with tests for software implementation errors and algorithmic errors using validation.

The scope of this work is the support of applications for Sentient Computing. A practical and pragmatic approach is adopted based on the current capabilities and performance of devices. The intent herein is not to build the most accurate or most highly performing (by some metric) system, but to construct systems with understandable, predictable behaviour whilst still meeting the goals of Sentient Computing.

1.1.1 Existing technologies

Chapter 2 presents a top-down survey of Sentient Computing. It begins with an examination of high-level application requirements, progressing on to the features and behaviours of location systems. Particular attention is paid to the reliability challenges presented at each stage.

By considering the needs of applications, this chapter aids the identification of design goals for dependable systems. Consideration of available technologies and current approaches motivates design choices made later in this work.

1.1.2 Engineering for dependability

Chapter 3 describes the provision of dependability by working upwards from the low-level sensors in a system. This is realised through the development of a dependability-oriented location system: the creation of Cantag, a marker-based machine-vision location system, is presented. Testing Cantag using an integrated test harness highlighted algorithmic problems with current marker-based vision techniques—these problems are rectified and the performance improvements are demonstrated.

1.1.3 Algorithmic dependability

Chapter 4 presents an investigation into the theoretical behaviour of Cantag. The goal is to define metrics which describe the current performance of the system. An information-theoretic argument is used to bound the best performance of this class of location system. The benefit of this high-level analysis is demonstrated through design-optimisation for circular marker tags.

Results from simulation are examined to compare the performance and stability of candidate image processing algorithms. Further metrics for system behaviour are developed.

Real-world results from Cantag are presented and compared to the simulated data. A dependable tag design and processing pipeline is identified whose simulated performance correctly predicts its real-world behaviour.

1.1.4 Runtime dependability

Chapter 5 investigates the concept of *validation* for data in a dependable system. Many of the computations in Cantag (and other Sentient Computing systems) are asymmetric. This means that the forward computation of the result from the input data is computationally more expensive than the backward computation required to check that the result is consistent with the input data. Validation is particularly useful in a dependable system to protect against implementation and algorithmic errors within the system.

Validation also integrates the previously identified metrics for system performance by using them as the basis for additional acceptance tests.

An example implementation of validation is presented and evaluated using logic programming (Prolog) extended by external predicates for interacting with the location system. The cost of validating particular pieces of context as required by an application is examined and techniques for reducing this cost are demonstrated.

Chapter 2

Related Work

2.1 Supporting Sentient Applications

The construction of a dependable application cannot proceed without first providing a dependable infrastructure. This infrastructure includes any sources of context used by the application, and additional support functionality such as facilities provided by a middleware.

This section presents various classifications of applications and sensor systems in order to codify the needs of sentient applications and the features provided by existing sentient infrastructure components. Location systems are of particular interest because they provide a primary source of context [40] for many applications and so are most likely to form a key part of future dependable systems.

2.1.1 Operating environment

Office environments have been a common focus for Sentient Computing. A telephone receptionist's aid based on room-level location was an early example [144]. More fine-grained location information has also been exploited in this environment to provide real-time maps and to automatically select cameras for videoing employees and visitors as they move around [145, Chapter 9].

Other targeted environments have included museums and exhibition halls in which researchers sought to provide additional information to visitors as they arrived at a particular exhibit [33]. Another study considered the hospital environment and provided a messaging system which can address users by role, location, and time [99].

Context-aware applications have also been deployed in the home using cues from the occupants' locations for smart control of media devices and lighting [84].

Outdoor applications include city-wide location-based gaming where virtual players are chased by runners in the real world [15]. Drishti is a navigation system for the blind which is suitable for both indoor and outdoor operation through handover between different location technologies [112].

Different environments present different challenges for a location system. Signal propagation is affected by the structure of the space: small offices limit propagation more than open-plan

spaces. Occupants' movement patterns differ between environments. In outdoor spaces people might move purposefully towards a destination or browse more casually whereas in office environments an occupant might primarily travel between their desk and the provided amenities. The acceptability of any location sensing is also environment dependent: users who participate at work may well be loathe to wear marker tags (or even to be tracked at all) at home.

Systems operating in outdoor environments must cope with extremes of temperature, lighting, and humidity produced whereas indoor environments benefit from some measure of protection from the weather. Also, useful infrastructure (such as power and network wiring) which might be exploited by a location system is more common within buildings rather than in outdoor environments.

2.1.2 Coverage and boundaries

Applications place widely ranging requirements on the coverage of a location system. The Invisible Train application uses a camera and a handheld computer (PDA) to present an augmented view of the world [141]. Only a limited coverage area is required because the acquired image is shown with annotations on a small PDA screen. Conversely, the ActiveMap application provides a visualisation of the (room-level) locations of users [95]. Coverage of a significant number of offices is necessary for this application to be useful.

A location system's boundaries are also an important consideration. A system is said to be *boundary transparent* if a user is not required to take any action when tracking commences. Users may enter and leave the coverage areas of boundary transparent systems without special action. Boundary transparency increases in importance as the amount of time the user spends within the system decreases. This is often correlated with coverage area: smaller coverage areas imply reduced residency times and thus increased requirement for boundary transparency.

The Polhemus Liberty tracker[1] is a magnetic tracker with very high accuracy over short ranges: $0.0038\,\mathrm{mm}$ RMS error within $30\,\mathrm{cm}$ of the field source. However, its use of a tethered stylus (which cannot leave the system at all) makes it boundary opaque.

The Active Badge system consists of mobile badges worn by users. The badges periodically broadcast a unique identifier over an infrared channel [144]. This is decoded by room-level receivers and passed to a central location service. Reflections from walls and furniture make the infrared signal likely to reach the receiver without requiring direct line-of-sight. This is an example of a boundary transparent system because the tags carried by the user are automatically acquired by the system when the user enters the room.

The camera-based tracking system used for the Invisible Train is boundary transparent because the tracked objects (fiducial marker tags) may enter and leave the field-of-view without hindrance. The application would become infeasible if this were not the case.

It is only acceptable for a location system to be boundary opaque if users' residency times within the system are of substantial duration.

[1] `http://www.polhemus.com/`

2.1.3 Types of location information

Applications typically expect a range of contextual data in addition to location information. These may include the identity of the locatable object and time of sighting. The form of these data vary according to the application's needs.

Desktop teleporting permits a user to move their desktop onto the currently visible machine [118]. This requires *containment* information i.e. which machine is contained within the spatial region which represents the user's field-of-view. The user's 'aura' is another commonly used term with this meaning. This type of information is referred to as *symbolic* information. Symbolic information has no spatial grounding. Interpretation of these symbols often requires a mapping from the symbolic identifier to the current frame of reference. For example, the symbolic identifiers of receivers in the Active Badge system are mapped onto the perimeter of the relevant room in the Euclidean space which describes the building.

Steerable interfaces have been considered by researchers as a means of intuitively positioning the interface to an application [108]. Examples include a reminder note application positioned on the desk of the user, and highlighting active real-world devices such as a ringing phone. These applications require a different kind of positioning information known as *metric* information.

The term metric information is very superficially related to a *metric space* in mathematics which defines a set where there is a notion of distance between the elements. Analogously, metric location information has the same property. The Active Bat system calculates location information from time-of-flight estimates of an ultrasonic pulse emitted by a mobile Bat [146]. This produces metric location information in a Euclidean coordinate frame. The Navstar (Navigation Satellite and Ranging) GPS (Global Positioning System) produces metric location in the WGS84 coordinate frame from the differences between arrival times of signals from time-synchronised satellites in known orbits [58].

The classification of symbolic or metric information is applicable to other forms of context provided by the location system. Systems such as the Active Badge system produce symbolic information for the identifier of the beacon and the receiver. The time of the sighting is metric information because the distance in time between two sightings is a useful concept. In the Active Bat system the identifier of the Bat itself is symbolic information whereas the time of the sighting, and the location and orientation of the Bat, are metric information.

A further example is the Active Floor which measures the Ground Reaction Force (GRF) at the corners of floor tiles using load sensors [1]. A Hidden Markov Model (HMM) is used to infer the identity of a walker through gait classification. The time of the measurement and location (in terms of which tiles are occupied) of the user are both metric information. In this case the identity of the user is also metric information. Classification of the user can be viewed as taking part in a space of classification features in which there are regions representing the learned classification for each individual user. The estimate of identity is a point within this space.

This final example exemplifies that the definition of distance in a metric system should contain some useful meaning. This is evidently true for coordinates and time. Furthermore, for the identity of a user in the Active Floor the similarity between users (in terms of gait or other features) is physically relevant. However, in the Active Bat system examining the similarity between the symbolic identifiers of two Bats is not particularly useful.

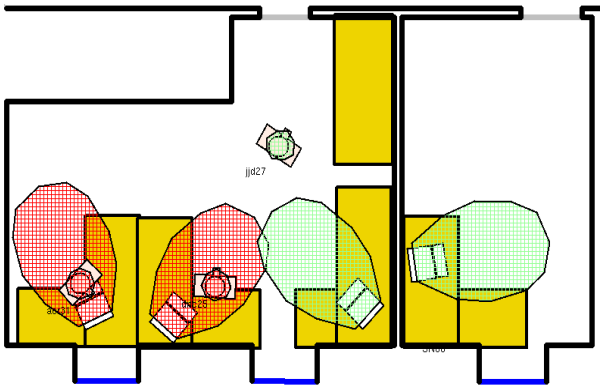


Figure 2.1: The SPIRIT map application showing *computer-visible* regions

In order to facilitate its application to quantities such as time, the term metric is preferred over alternatives in the literature which suggest a spatial interpretation such as Coordinate [46] or Geometric [91].

A common function of a middleware is to translate between types of location information. An example of this is the SPIRIT middleware which generates containment information from the coordinates of tracked Bats [2]. Figure 2.1 shows a screen shot of the SPIRIT map application. The *computer-visible* regions are shown and triggered interactions with two users are highlighted.

Translation of contextual information is obviously limited by the granularity of the underlying information. Location systems providing high-precision (fine-grained) information should be able to support more applications than those systems with lower precision information.

2.1.4 Information delivery

Applications may be classified as either event-based or information-polling. An event-based application is notified whenever a particular type of event occurs. An example of this is the Active Poster application [2] which triggers an event when the button on an Active Bat is clicked in the region of space covered by the poster. Other examples of event-based applications include desktop teleporting [118]; the Forget-me-not project which records events such as the interaction of two users or entry into a room [86]; and location-aware museum guides which display relevant information when entering a new area [33].

Information-polling applications acquire contextual data on demand. Semantically, this is requesting the *last-known state* of an entity. Simple examples of this are requests such as “Where is Andy?”. A more complex example from the Active Bat system is the use of raw sighting information to determine the location of untagged items (e.g. computer monitors and office partitions) for maintenance of the world-model [61]. This application requests and processes the substantial amount of raw sighting data for the spatial volume of interest.

It is possible to construct information-polling applications with an event-based approach simply by collecting all possible events pertaining to data which may be of interest in future. This is not possible for applications with a large set of potentially relevant information running on

impoverished devices. However, more powerful devices can use this approach to provide a information-polling service to applications which require it.

2.1.5 Direction of observation

Welch and Foxlin classify location systems as either *outside-in* or *inside-out* [149]. Outside-in systems accumulate the necessary information for location tracking from a fixed infrastructure. In this case sensors are *looking in* at locatable objects. The Active Badge system is an example of an outside-in system.

Conversely, an inside-out system permits the tracked device to sense information for estimating its own location. An example of this is The Locust Swarm. In this system solar-powered, infrared beacons (Locusts) are deployed into the environment (typically under fluorescent lights) [129]. The beacons repeatedly broadcast a preset location code which is interpreted by mobile devices passing under the beacon. Each locust covers a region of up to six metres in diameter.

As can be seen from the above examples, some location techniques can be applied in either orientation: Active Badges broadcast identifiers which are received by fixed infrastructure whereas fixed Locusts broadcast region identifiers which are collected by mobile nodes. Typically, outside-in systems benefit from increased computational and power resources for the post-processing of sensor information as compared to inside-out systems which must perform this operation on a mobile, lower-powered node.

The Cricket system is an inside-out system operating with fixed ultrasonic beacons from which a mobile tag resolves its position [109]. The system was designed to preserve the privacy of users within the system because only the mobile device (and not the infrastructure) knows its location. For this reason, inside-out systems are occasionally termed *privacy-oriented*.

2.1.6 Search constraints

A further axis of classification of a location system is whether the system is *tagged* or *tagless*. A tagged system tracks artificial markers deployed in the environment or attached to users whereas a tagless system produces information from an unconstrained scene. This distinction is relevant to applications because the identity of the locatable object reported by a tagless system commonly has uncertainty in it due to similarities between objects. The identity information provided by a tagged system is often less uncertain in this respect but the different consideration of whether the tag is actually attached to the correct entity is now pertinent.

Tagged systems are popular because the design and behaviour of the tags simplifies the location process. For example, in the Active Bat system the mobile Bats broadcast ultrasonic pulses with a known profile only when polled. This controls interference on the ultrasonic channel and permits direct estimates of the time-of-flight of the received signal.

Tagless systems operate by exploiting natural features of the scene. This precludes the use of sensing signals not naturally emitted by entities to be tracked. An ultrasonic system suffers in this way because people (the common target of a location system) provide few natural features for an ultrasonic sensor. However, tagless localisation of audible sounds using microphone

arrays is a popular topic. Bian et al. use auto-correlation between pairs of microphones to estimate the difference in distance travelled by the signal followed by a steepest gradient search for a location consistent with these differences [19]. Scott and Dragovic utilise a non-linear least-squares regression to directly solve a system of equations containing the (unknown) sound source and the (unknown) time of emission [125]. These systems focus on localisation rather than classification of sounds or idenfification of users.

The EasyLiving project [84] is an example of a tagless machine vision system. The tracking system is composed of a number of stereo camera-pairs which locate people moving in the scene. A colour histogram technique is used for identification which suffers from occasional ambiguities when users wear similarly coloured outfits—the concept of a metric identity axis is useful here for comparison between users with similar appearance.

Tagless systems often operate within a poorly defined space of true positive sightings. The complete set of distinct entities requiring identification is often unknown and thus the minimum "distance" between two distinct entities can only be estimated. For audible sound location systems the variation in sounds which construe an event of interest is huge (consider the difference between a conversation and applause). Tagged systems have foreknowledge of all the parameters of the tracked tags at the design stage and therefore constrain both the size of the search space and the similarity of distinct entities. It is sometimes possible to estimate the distribution of positive sightings in tagless systems. An example of this is given by Daugman and Downing for human iris patterns [34]. Their analysis, using quadrature wavelets, found 244 independent degrees of freedom between pairs from a sample set of 2.3 million images of human irises.

Tagged systems also benefit from reduced computation costs compared to tagless operation. This contrast is particularly marked for machine vision systems. Implementations make use of *fiducial markers* to assist the tracking process. The markers can provide geometric invariants to assist in the location process and coded payloads for robust identification. Particular examples of this are the Matrix system [114] which tracks square tags and the TRIP system [37] which tracks circular tags. These systems generally produce more reliable identification and vastly improved tracking rates than tagless systems. The EasyLiving tagless system produces sightings at $3.5\,\mathrm{Hz}$ running on PC hardware whereas the tagged vision system SpotCode runs at $15\,\mathrm{Hz}$ on a mobile telephone [124].

2.1.7 Statefulness

When considering digital electronics, sequential logic circuits with feedback are substantially harder to analyse than those without it. This is because feedback creates state—the current output of the component has some effect on the next output. Similarly, a location system whose current output has no effect on subsequent outputs (i.e. without feedback) should be more tractable to reason about than a system where historical data impacts upon the current value. Systems such as the Active Badge or Bat systems are stateless. In these systems the sensor reading(s) which contribute to a particular location sighting are discarded when new data arrive.

Fox et al. discuss the use of Bayesian filters for location estimation [54]. They use a selection of filters to fuse historical data from various sensors into an estimate of the current position. This permits new estimates of location to be based upon previous estimates and can prevent errors such as implausible translocation of a user. These filters create a stateful system because the

output of the system for each new input value is dependent upon the previous input values as well.

Lee and Mase present a stateful system which utilises a two-axis accelerometer and a digital compass to match a movement trace against a database of location transition traces and thus estimate the resulting location [88]. To improve accuracy their system also makes use of the previous estimate of location when selecting matches for the transition trace.

Stateful systems are capable of producing more reliable and more accurate location information than a stateless system because occasional estimation errors are absorbed by the historical data. However, many of the filters utilised require a priori estimates of error probabilities and the question still remains of why these errors occur in the first place. A complete understanding of the stateful behaviour depends upon an understanding of the stateless systems underlying the filter.

2.1.8 Sensing interface

Many sensing mediums are available to form the basis of a location system. The requirement for unobtrusive technology means that the sensing technique must be imperceptible to the user. For dependability, the behaviour of the transmission medium must be predictable on the same granularity as the produced location information.

Ground Reaction Force

Physical systems rely on transmission of force between the tracked user and the sensor. Most commonly these systems are floor-based in order to ease this transfer.

An extensive survey of sensing requirements and techniques for plantar (relating to the sole of the foot) sensing is given by Urry [138]. Pertinent details are: for recovering information about the human foot and gait a precision of about five millimetres gives maximal information. Signals with frequencies as high as $75\,\mathrm{Hz}$ have been observed in footfall impacts of the heel. A sensing range of 0 to $10^6\,\mathrm{N/m^2}$ is appropriate for tracking a walking person.

Location systems will be subjected to larger extremes of force than plantar measurement systems. High-heeled shoes (not contemplated in plantar sensing) are particularly problematic: a person with a mass of $65\,\mathrm{kg}$ exerts a pressure of over $3 \times 10^6\,\mathrm{N/m^2}$ through a high-heel of size $2\,\mathrm{cm^2}$. The instantaneous pressure when walking will be higher than this. Many of the techniques in plantar sensing such as Force-Sensitive Resistor (FSR) arrays, capacitance mats and piezoelectric plates will break under this level of pressure.

In light of these problems, and cost considerations for wide-scale deployment, the Active Floor used load sensors in the corners of $50\,\mathrm{cm}$ square floor tiles [1]. Each sensor produces a reading representing the load on the four adjacent tiles whose corners rest on it. The load cells have a rated load of $5000\,\mathrm{N}$ and an error of $\pm 25\,\mathrm{N}$. The researchers observed that the majority of the data signal from the load sensors lay under $250\,\mathrm{Hz}$ for people walking and running over the floor. This value is higher than suggested through plantar sensing [138] but can perhaps be explained by vibration of resonant parts in the floor structure itself. The Active Floor identified walkers through gait classification with a Hidden Markov Model. An alternative approach, used in the

Smart Floor [104] created a ten-dimensional feature vector from the trace containing values such as maximum load cell response from the heel strike and toe push-off. Both techniques achieved a false-classification rate of around 10%. Although it is expected that individuals have highly distinctive foot-pressure patterns when measured at high resolution [138] it remains to be seen whether low resolution sensing is sensitive enough to distinguish between large sets of individuals.

Other projects attempting to build more precise floor sensing have mostly targeted interactive dance. The Magic Carpet uses a grid of piezoelectric wires spaced approximately ten centimetres apart [107]. The grid layout means that this approach is unable to distinguish correct positions for simultaneous footfalls. The Pressure Sensing Floor [128] uses FSR arrays from Tekscan[2] to achieve tracking with $6\,\mathrm{mm}$ precision. Both of these projects sacrifice the robustness of the sensors under high pressures to improve precision.

Floor sensing systems can only recover location information in two dimensions and cannot capture movements (such as small hand gestures) which do not change the GRF. Path analysis and the use of home-locations (a region of space commonly used only by a single user) could be used to provide additional information for inferring identity. The huge range of forces experienced by flooring makes fine-grained sensing particularly demanding. Existing fine-grained solutions have limited robustness in this respect.

Inertial measurement

Accelerometers and gyroscopes form the basis of inertial sensing. An accelerometer measures acceleration along a particular axis. Accelerometers are often combined in packages with two (or three) orthogonal axes in order to cover all of two-dimensional (or three-dimensional) motion. In order to derive location information the acceleration value is integrated twice to yield a position. The constant values introduced by the integration represent the fact that starting point (and velocity) of the path is unknown.

A gyroscope consists of a spinning rotor mounted in a mechanism (a gimbal) which permits free rotation of the axle. The rotor rolls with changes in orientation to retain its original aspect. This provides a reference direction with which to measure orientation.

The InertiaCube[3] combines accelerometers, gyroscopes, and magnetometers (for sensing changes relative to the Earth's magnetic field). It provides roll, pitch, and yaw information at an update rate of $180\,\mathrm{Hz}$ with a error of approximately $1\,^\circ$. In the NavShoe project an InertiaCube was mounted on a shoe to measure the walking movement of a pedestrian [55]. One particular source of error in this system is the effect of accumulated drift in the angle reported by the gyroscope. This is important because the accelerometers are continually experiencing an acceleration due to the Earth's gravitational field. The measured effect of the gravitational force depends on the current orientation of the accelerometer's sensing axis to the gravitational field. The correction factor is calculated from an estimate, provided by a gyroscope, of the absolute orientation of the accelerometer. Error in the estimated angle of the accelerometer axis introduces errors in this compensation factor—for small angles of inclination (perpendicular to the direction of gravity) the remaining error in the acceleration reading is roughly linear to the error

[2] `http://www.tekscan.com/flexiforce/flexiforce.html`
[3] `http://www.isense.com/products/prec/ic3/wirelessic3.htm`

in the angle estimate. The acceleration values are double integrated to produce the location and so the location error grows cubically with the gyroscope errors which grow linearly with time. However, identifying the stationary stance phase in the tracked person's gait permits the injection of a correction factor—the velocity of the foot is known to be zero at this time. This factor anchors cubic error growth to the beginning of the stride and so the resulting error rate is linear in the number of strides. This technique allowed the NavShoe tracker to achieve an accuracy of approximately 0.3% of the total distance travelled. This error rate can become significant over time: it is not unusual to imagine an individual walking over a kilometre in one day. This corresponds to an error of greater than three metres.

Fawcett gives a summary of the sources of error in gyroscopes and accelerometers [46]. Acknowledgement of these sources of error permit inertial sensing instruments to be constructed and calibrated to high tolerances. Additionally, the NavShoe project demonstrates the importance of domain specific corrections. A high quality sensor with cubic error growth will rapidly become more inaccurate than a low quality sensor with linear error growth.

The high update rates of inertial trackers can also be exploited to assist slower, but more accurate, systems in the tracking process. The VIS-Tracker is a machine vision-based location system which is capable of producing globally accurate location readings of deployed marker tags [100]. An InertiaCube is integrated into the system in order to direct the search area of the image recognition process so that acquired tags can be tracked without rescanning the entire image.

Inertial sensors must be physically attached to the tracked object. Common garb and adornments such as shoes and wrist-watches mean that this is not particularly disruptive to users. The update rate of these sensors is typically an order of magnitude higher than other location sensing technologies. Absolute reference points are vital for inertial systems because even small rates of error can accrue to significant position errors. The acquisition of these reference points is conventionally through some other sensing mechanism—such as a machine vision system in the VIS-Tracker.

Ultrasonic sensing

Sound waves have a propagation speed in air orders of magnitude slower than electromagnetic waves: the speed of sound in air at room temperature is approximately $340\,\mathrm{m/s}$ whereas the speed of light is approximately $300 \times 10^6\,\mathrm{m/s}$. The slower speed of sound makes it amenable to timing and hence range estimation. Sound frequencies just above the human hearing range are normally utilised (around $20\,\mathrm{kHz}$). This ensures unobtrusive operation of the system and minimises the signal degradation through absorption in air, which increases with frequency [13]. In an indoor environment ultrasonic waves can be considered to travel in a straight line. Effects such as diffraction around objects or the Doppler effect from objects in motion [121, Chapter 3] are often assumed negligible.

The propagation speed of sound through air has been extensively investigated [22]. The primary factor affecting the speed of sound is the air temperature. Humidity is also important but to a significantly lesser extent. The Active Bat system includes a temperature sensor in each room and incorporates a compensation factor in its distance calculations [145, p35]. This is sufficient for 95% of the positions produced by the system to be within $3\,\mathrm{cm}$ of the true location [2].

A similar approach in the Cricket location system integrated a temperature sensor with every ultrasonic beacon. The temperature reading is included with each broadcast from the beacon to permit correction by the mobile device. Cricket achieved an accuracy of $10\,\mathrm{cm}$ [111, p96]. Other ultrasound localisation techniques include the temperature as an unknown in the system of simultaneous equations [48].

The Active Bat system can also produce estimates of orientation information from a single transmitter by exploiting the shadowing effect of the wearer's body. This causes a directional pattern in the ceiling sensors which receive the ultrasonic pulse. This estimate is within 60° for 95% of positions [145, p75].

The relative position of multiple transceivers can provide more fine-grained orientation information. The Cricket Compass system utilises a "V" shaped array a few centimetres in size containing five ultrasonic transceivers for this purpose [110]. The resultant orientation accuracy has an error of the order of five degrees.

There are numerous sources of ultrasound in an office environment which interfere with ultrasonic location systems. Particular examples are the electric motors in vacuum cleaners, jingling keys, and key presses on some types of computer keyboard. The Dolphin system alleviates this problem through the use of broadband ultrasonic transceivers which enables reliable data communication over the ultrasonic channel [66]. The system is bidirectional and so can operate in centralised mode or a privacy-oriented mode. Location errors are within $2.2\,\mathrm{cm}$ for 95% of sightings in centralised operation and within $5\,\mathrm{cm}$ for privacy-oriented operation.

Smooth, planar surfaces such as walls and tables reflect ultrasonic signals with little attenuation. Therefore, sensors are likely to see a large number of reflected signals. Typically, these signals are detectable as inconsistencies in an over-constrained dataset. The iterative non-linear regression (INLR) algorithm [145, pp. 36–39] used in the Active Bat system attempts to deal with this problem by repeatedly hypothesising (using non-linear regression) a best-fit location and discarding significant outlier points. The Random Sample Consensus (RANSAC) algorithm is an alternative to this approach [49] which derives a position from a randomly selected triplet within the data set. The remainder of the data set is then partitioned into supporting and non-supporting data based on the expected error of each reading. If a quorum of suitable size is found the algorithm returns the estimated position. Otherwise, another triplet is selected randomly and the process repeated until a predetermined number of iterations has elapsed. A comparison of these approaches with real data shows the superiority of the INLR algorithm [117]. However, this algorithm can be misled in the case when the majority of signals arrive through the same reflected path. Figure 2.2 shows the error in the position of an Active Bat relative to its surveyed (by theodolite) location. The Bat is fixed to a wall of a room and the second peak in the frequency diagram corresponds to situations where signals reflected off the wall have formed the result returned by the multilateration algorithm.

Most ultrasonic transducers utilise the piezoelectric effect. Piezo-ceramic transducers (as used in the Active Bat system) benefit from low cost and high durability but have a low bandwidth range in the emitted signal (less than $5\,\mathrm{kHz}$). Wideband (greater than $60\,\mathrm{kHz}$) transducers (as used in Dolphin) can be constructed from piezo-polymers at the cost of increased complexity in the electrical interface to the material [65].

Ultrasonic location can also be performed using commodity hardware. The WALRUS system uses the microphone built into a PDA to detect ultrasonic beacons to derive room scale information [23]. The room identifier and other metadata are broadcast over a wireless 802.11b

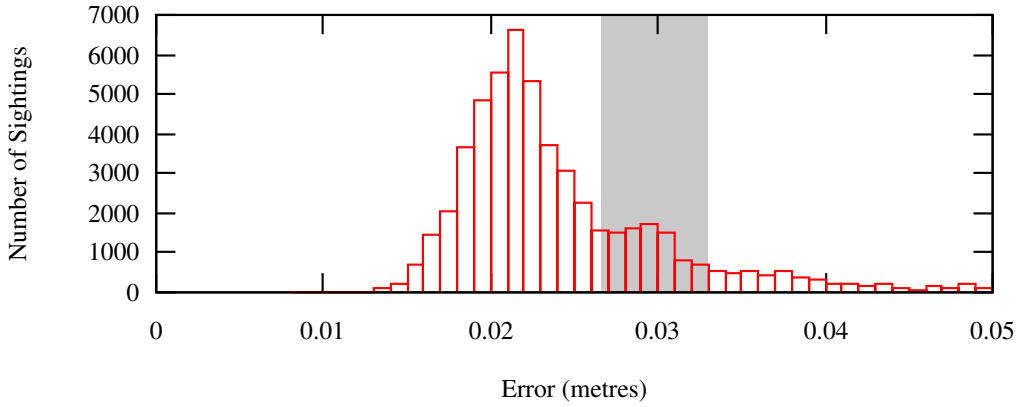


Figure 2.2: Position error in the Active Bat system due to multipath signals

network. If the device receives an ultrasonic pulse following the broadcast it infers that the broadcast pertains to the present room. Due to the uncoded nature of the ultrasonic pulse it is expected that false sightings will be commonly triggered by other sources of ultrasound in the environment.

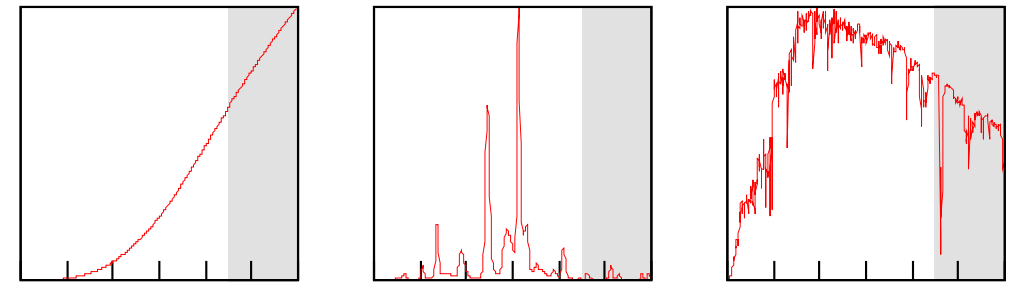
In summary, the propagation of ultrasonic signals intuitively conforms well with physical partitions in the environment because the signals will not penetrate walls or floors. The relatively slow speed of sound through air enables the use of signal timing techniques for location and effects on propagation speed (such as air temperature) can be measured and compensated for. Low accuracy systems have been produced using commodity devices. More accurate ultrasonic positioning has been achieved through the use of specialist hardware.

Infrared sensing

(Near-)Infrared signals have a number of appealing characteristics for the design of a location system. Their propagation is similar to that of visible light and so intuitively conforms to barriers (such as walls and partitions) in the physical environment. Also, by virtue of the limits on human visual acuity, infrared signalling is unobtrusive.

Systems such as the Active Badge and Locust Swarm make use of photodiodes for sensing and emitting infrared signals. The infrared data-channel was originally used solely to transmit the identifier of the Badge. In subsequent revisions of the Active Badge additional functionality was added to include challenge and response authentication of the Badge and to play sounds for audible feedback [63]. The ParcTab system evinces an extension to this functionality with a touch-sensitive display, buttons and a loud-speaker [142]. ParcTab includes general data communication over the infrared channel to allow thin-client operation. More recently, the CoolTown project utilised the infrared port common on many handheld PDAs to broadcast location triggered information [81] such as URLs.

Photodiodes are a particularly reliable form of sensor. They provide a low noise, highly linear response with respect to the intensity of the measured light and have a wide spectral response. The difficulty experienced with infrared systems is the high level of background noise present in the environment. Figure 2.3 shows the background noise created by some common lighting techniques. The near-infrared signals used by these systems are around $750–900\,\mathrm{nm}$ wave-



(a) Incandescent Bulb 60W (b) Fluorescent Tube 36W (c) Solar Irradiance [10]

Figure 2.3: Relative spectral emissions from selected light sources. The near-infrared band is highlighted

length. Direct sunlight will commonly overwhelm an infrared signal. One observation that can be made from these curves is that the use of fluorescent tubes to provide the solar energy for the Locusts in the Locust Swarm was a good choice as opposed to incandescent lighting which would have caused significant interference.

More sophisticated infrared sensing systems also exist. The Irisys tracker uses a 16×16 element sensor to track moving people[4]. Despite the low granularity of the sensor this system is capable of discerning the paths of individual users and has many current applications including customer tracking and security systems.

In addition to coping with large background noise levels another potential source of failure in infrared systems is that of uncertainty in the implicit containment area defined by the infrared propagation distance. Systems such as the Active Badge aim to beacon with sufficient intensity such that the signal reaches the receiver regardless of the badge's position in the room. However, in open-plan areas, or occasionally in rooms with open doors this signal can propagate to an adjacent reader causing an erroneous location sighting.

Infrared location systems can be cheaply constructed and are unobtrusive in operation. However, the location information produced is often only proximity information because it is based on the identifiers of those signals which can be successfully decoded.

Vision systems

Machine vision systems are a popular choice for sensing location because cameras for image capture are cheap and ubiquitous—especially given their recent appearance on mobile telephones. Another important benefit is that visible light is directly experienced by (most) users of the system. Thus, a comfortable environment tends to be well illuminated. The transmission of visible light is intuitively understood by users of the system and so the propagation restrictions (such as a requirement for line-of-sight) are easy to comprehend.

Charge Coupled Device (CCD) arrays form the basis of the majority of capture devices. A CCD array consists of an array of capacitors (buckets) which accumulate charge when struck

[4] `http://www.irisys.co.uk/products/smartcounters.htm`

by photons. The accumulated charge is then read out sequentially by passing it from bucket to bucket. As an alternative CMOS arrays are now a viable alternative due to the continuing improvements in VLSI fabrication. Extra transistors are used in these arrays to allow direct reading of each pixel—a CCD array must read out the entire image each frame whereas a CMOS array may focus its area of interest.

A key factor influencing the accuracy of any information derived from the image is the effect of the lensing system. Lenses can be precisely modelled and calibrated [26]. Systems requiring accurate measurements from images utilise static calibration information from the lens to apply distortion corrections.

The colour histogram technique used to identify users in the EasyLiving project is also exploited in other work to identify locations. Starner et al. derive a feature vector of colour intensities from forward facing and downward facing cameras [130]. Colour intensities from an image of the user's nose are also collected to be used as a means for detecting the lighting conditions in the room. This has the affect of applying a correction for the current lighting conditions affecting the values from the first two cameras. A Hidden Markov Model is used to estimate the room level location of the user. Aoki et al. attempt to classify locations using sets of colour histograms as feature vectors representing the user's trajectory [9]. This additional state should provide additional information to help disambiguate rooms. These systems suffer from a poorly defined search space: the range of inputs for a particular room is difficult to estimate because of factors such as lighting variation and large number of possible locations.

A further system, designed for robot localisation, estimates position from a single intensity value due to the lighting above the robot. This is estimated from a small window in the centre of the acquired image [39]. A particle filter (known as the Condensation algorithm) is used to fuse inertial navigation data with this intensity value with reference to a pre-generated intensity map of the ceiling. Application of this technique enhances the "hopelessly" inaccurate navigation process to permit the robot to navigate to within $10\,\mathrm{cm}$ of the target position.

Localisation through the Condensation algorithm without inertial navigation input is attempted by Rungsarityotin and Starner using an omni-directional camera [122]. The omni-directional camera is exploited by assuming that images contain negligible translation relative to the training set and so an orientation invariant comparison is devised by comparing rotations of the captured image to the training data. The assumption of negligible translation only holds if the tracked user remains close to the original training path. The results from this approach seem to suggest successful tracking of particular movement sequences but only from favourable starting conditions.

Augmented Reality relies on video overlay techniques to superimpose virtual artefacts over a viewed image. Thus, registration of the physical and virtual world is vital for the correct placement of the overlay. Maintaining this registration through cues in the image itself simplifies the problem because there is no need to calibrate the location system with the camera. Klein and Drummond describe a system for tracking and visual overlay which matches a CAD model of the environment to edges occurring in the scene [82]. The authors note that the system requires uniquely distinguishable objects and the reliance on a CAD model of edges precludes support for natural features such as trees or constantly changing environments.

The use of fiducial markers in tagged vision systems enforces constraints on the scene which guarantee invariants for tracking. Two-dimensional, passive tags are common which can be

printed cheaply with commodity printers. Systems such as Matrix [114] and Cybercode [115] recover position information from square tags. Alternatives such as TRIP [37] and the VIS-tracker [100] utilise circular tags. The use of fiducial markers permits stronger guarantees about object recognition than for unconstrained systems because the search space is better defined. Furthermore, these markers provide geometric invariants which allow the system to recover three-dimensional position and pose from a single camera. This capability provides similar information to Drummond's CAD model approach but has a fixed structure rather than requiring a new model for each object.

Position estimation can be also be achieved though simultaneous tracking of multiple tags. The Free-D system [136] deduces the location of a mobile camera from the intersection of rays through the centre of multiple tags. This technique is more generally known (notably in the field of photogrammetry) as bundle adjustment.

In order to further reduce computational cost, or to improve update rates, fiducial marker systems often utilise inter-frame state to constrain the search space of the tracking process. For example, TRIP searches for image features proximate to previously recognised tags. The VIS-tracker integrates information from inertial sensors (the InertiaCube) to direct the search space.

Manual, user-reported location systems might also be classified as vision systems. One investigation of user-reported location in a location-based game found the median error of such declarations to be approximately $25\,\mathrm{m}$ [16]. Commonly self-reported location will not suffice for the ideals of Sentient Computing because requiring a user to continually state his position will increase, rather than reduce, the cognitive load of the system.

Vision systems can produce high-precision information from commodity hardware and the use of visible light is intuitive to most users of the system. Vision systems are often very sensitive to lighting conditions and scenes are often visually cluttered. Marker tags can be used to improve the performance of the vision process. Vision systems also have the capability to gather contextual information other than location such as the current emotions of the user from their facial expression [79].

RF systems

Signal strength is one common means for deriving location information from Radio Frequency signals. The simplest systems are proximity systems which report a sighting when the signal strength is sufficiently high for communication or identification to occur.

The Active Badge system implemented an early example of a radio proximity system. Radio antennae deployed in the environment were used to broadcast an identifier detectable by a tuned coil in the Badge. The received identifier was subsequently broadcast over the infrared channel along with the identifier for the Badge itself [62].

Passive Radio Frequency IDentification (RFID) tags are becoming increasingly ubiquitous. The appeal of these systems is that the tags do not require an integrated power source but instead draw power parasitically from the radio field of the tag reader—this permits smaller, longer lived, more reliable designs than possible for active tags (with their own power source). Want et al. utilised RFID tags to augment real-world objects for reading with a mobile reader [143]. They identified a number of trade-offs when using RFID tagging: the tags are small, unobtrusive, robust, and easily sensed but this creates administrative problems in associating the tag

identifier with the correct action (a problem common to all tag-based systems). The unobtrusive nature of the tagging also creates difficulties in knowing which objects are tagged.

Uncertainty in RFID systems comes from a number of sources. Important sources with respect to a location system are power scavenging from the reader's radio field, and interference with other tags. Radio field strength propagation is highly complex, particularly in indoor environments [87]. This makes the size and shape of the spatial container covered by the RFID reader complex and variable over time. Various protocols exist for coping with collisions between multiple tags [77] including Ethernet style back-offs and binary-search addressing algorithms. These algorithms feature different trade-offs in terms of acquisition time, maximum tag densities, and computation power on the tag and the reader.

Brusey et al. attempt to alleviate the rate of false negative sightings (radio propagation problems) and false positive sightings (collision problems) by utilising time-weighted averaging of sightings [27]. This decreases the false sighting rate at the expense of increasing the response time of the system. More fine-grained information is difficult to recover because conventional RFID readers do not report signal strength for the identified tag. Researchers have instead utilised the sighting rate of a tag as an estimate of its proximity [50]. Attaching multiple tags to an object then permits the detection of additional features (such as rotation).

The location information gained from these systems is simply the symbolic identifiers of the entities within the reading range. More fine grained information can be derived by examining the intensity of the received signal. Received signal strengths in indoor environments are affected by both large- and small-scale fading [87]. Large-scale fading is caused by the high attenuation of the signal through walls and floors. This attenuation is dependent upon the quantity and type of materials penetrated. Small-scale fading is caused by the superposition of the numerous reflected, diffracted, and scattered signals which reach the receiver. Small changes (such as opening or closing of interior doors) have a considerable effect. Location systems in this space generally apply *empirical* signal propagation models. The term empirical is used because these models are parameterised on some measured signal-strengths at known positions in the environment.

The RADAR system attempts to derive fine-grained location information from the received signal strength of multiple WaveLAN base-stations [12]. The training data collected include the orientation of the user because the obstruction has a significant effect on the received signal strength. A number of techniques for position estimation are proposed. The first method returns the best matching reference position and achieved an accuracy of better than $3\,\mathrm{m}$ (median error). In the second method the reference positions are numerically generated using an attenuation model which considers the number of partitions the signal has passed through. The reported accuracy of this technique was $4.3\,\mathrm{m}$ (median error). Further studies of the RADAR system confirm these accuracy readings and introduce a new matching technique additionally incorporating historical signal strength data from the mobile station [11]. The accuracy is increased to $2.37\,\mathrm{m}$ (median error).

An alternative approach was described by Smailagic and Kogan wherein a simple radial model is used to relate signal strength to distance. Lateration is used to estimate a position in "signal space" from the distance measurements. This position is then mapped into "physical space" using a set of translation vectors derived from training data. An accuracy of $2\,\mathrm{m}$ (median error) was achieved. For comparison purposes the best match algorithm from RADAR was also tested and achieved an accuracy of $4\,\mathrm{m}$ (median error) as opposed to the $3\,\mathrm{m}$ figure originally reported.

Assuming sound experimental method from both investigators this discrepancy must be put down to variations in the signal propagation environment between the two test sites.

These approaches are all limited in accuracy because of the granularity of the modelling technique. More advanced models, known as deterministic models, utilise techniques such as ray tracing and formulae for diffraction and scattering to produce more detailed models [87]. However, the significant effects caused by movement of doors and furniture limit the accuracy available. Investigation into temporal fluctuations in the channel for the purposes of wireless network provisioning model these effects with stochastic models of typical human movements [85].

One technique for improving the reliability of the modelling is to deploy fixed markers in the environment. These markers can be used to frequently measure the signal strength and provide new parameters for the empirical model used. The Landmarc system deployed static reference tags at known locations. The location of a mobile tag was derived from the set of proximate reference tags [102]. This approach achieves a median error of around $1\,\mathrm{m}$. This approach mitigates the problem of unpredictable changes in the propagation environment. Instead designers must consider the problem of deploying sufficient marker tags and ensuring that they remain stationary.

Deriving location from commodity wireless networking hardware is appealing from a cost and deployment standpoint. Bluetooth is another networking technology with growing ubiquity (particularly in mobile telephones). Feldmann et al. describe a location system utilising the Received Signal Strength Indicator (RSSI) to estimate the distance from the base-station [47]. They achieved an accuracy of $2\,\mathrm{m}$ but the coverage of the system is limited to within $8\,\mathrm{m}$ of the access points. Madhavapeddy and Tse attempted to characterise the propagation characteristics of a single Bluetooth transmitter by accumulating a large number of signal strength readings correlated with position from the Active Bat system [93]. They conclude that Bluetooth is ill-suited to accurate, low-latency location because of poor hardware support (signal strength is often only available as a running average), high variation in the recorded signal strength for large ($8\,\mathrm{m}$) distances from the receiver and the inability of many devices to maintain more than one Bluetooth connection simultaneously (this rules out triangulation and other techniques).

Time-of-flight measurements are an alternative to received signal strength for estimating the distance a signal has travelled. The Navstar GPS is an example of this mode of operation [58]. GPS signals are typically not strong enough for use in indoor positioning although future enhancements and the European GALILEO positioning system[5] will improve this [38].

Ultra-wideband (UWB) radio [17] has recently begun to be applied to location tracking. This technology alleviates the problems caused by destructive interference of multipath (reflected) signals in conventional radio systems. Conventionally, reflected signals can destructively combine with the direct-path signal. This makes the direct-path signal difficult, and sometimes impossible, to detect. UWB radio benefits from extremely narrow pulse widths (of the order of $2.5\,\mathrm{ns}$). As long as the difference in time-of-flight from the direct path signal and any reflected signals is greater than this (corresponding to a distance of about 10cm) the signal can be reliably distinguished. Fontana described a location system built on this technology which achieved accuracies of $30\,\mathrm{cm}$ [51]. Equivalent performance is available in a commercial system developed by UbiSense[6].

[5] `http://ec.europa.eu/comm/space/programmes/galileo_en.html`
[6] `http://ubisense.net/`

Small-scale fading effects make the use of radio location systems in indoor environments particularly difficult. Small changes in the environment can cause large changes in the received signal strength. The use of UWB radio makes signal-timing approaches feasible within indoor environments but requires specialist, high-speed sensing hardware.

2.1.9 Calibration

Commonly the model of the physical structure and behaviour of the system will vary between deployments. Calibration is the parametrisation of this model.

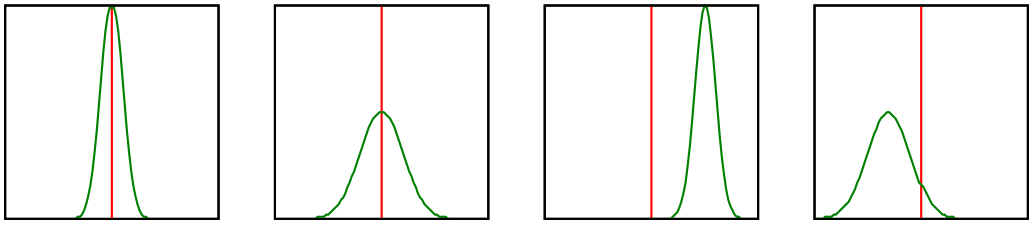
The construction of a two axis accelerometer aims to provide two orthogonal axes but due to slight imperfections in the manufacturing process or materials used these axes will slightly overlap. The calibration of this device calculates the corrections required to compensate for this effect. Any error in the calibration of the device will cause errors in addition to those due to inaccuracies of the sensing medium.

Another example of calibration to remove systematic errors occurs in CCD sensor arrays. Slight imperfections in the manufacturing process cause the sensing devices in the array to have slightly varying quantum efficiency (conversion rate of photons to electrons). These variations cause systematic errors in the collected data. Repeatedly reflecting a nearly uniform light source using a nearly uniform reflectance card onto the sensor provides sufficient information for deriving the sensitivity of each pixel [68].

A source of calibration error in time-of-flight location systems is in the location of the static reference points (whether they are receivers or transmitters). Initial deployments of the Active Bat system made use of a mechanical measurement system to determine the positions of the receivers in the ceiling array. This technique has now been replaced with the use of laser surveying equipment (a theodolite) which provides higher precision and more accurate calibration information. However, initial deployment of the system remains time-consuming and recalibration is required if ceiling receivers are subsequently moved. Until recalibration is performed, system accuracy is degraded in the region of the receiver.

To ameliorate this problem researchers have investigated the possibility of self-calibrating systems. Minami et al. describe an ultrasonic location system which iteratively locates the sensors in an un-calibrated sensor array from three initial seed-points [98]. This approach suffers from accumulation of errors because the error in position of each new sensor adds to the error in position of the reference sensors. Additional refinements to the system to aid in the selection of sensors with good position estimates provided an error rate of less than $5\,\mathrm{m}$ (95% confidence interval).

Various options for automatic calibration of the ceiling arrays in the Active Bat system have been studied [126]. The most successful of these was to use a simulated annealing approach to refine the estimates of sensor position using distance readings from a fixed array of Bats placed in various positions in the room. The positions of the sensors were learnt up to the accuracy of the Active Bat system i.e. 95% of sensor locations were within $3\,\mathrm{cm}$.



(a) High accuracy, High precision

(b) High accuracy, Low precision

(c) Low accuracy, High precision

(d) Low accuracy, Low precision

Figure 2.4: Accuracy and precision of location estimates

2.2 Uncertainty in Location Information

A low granularity approach [70] to modelling uncertainty information is to classify information in the following manner:

- **Sensed** information comes from the location system and is labelled as possibly inaccurate and out-of-date.
- **Static** information is initially entered by administrators and never changes. This information is considered highly accurate.
- **Profiled** information is provided by users and may vary over time. It is usually accurate when entered but is prone to staleness.
- **Derived** information arises from the combination of some number of the first three classes and inherits the worst of their characteristics.

Accuracy and precision can be used for a more precise specification of uncertainty in sensed information. The accuracy of a system refers to the discrepancy between the reported value and the true value. The precision of a system refers to the deviation of reported values from the *sample* mean (the mean of the reported values) [90, pp. 95–96]. Figure 2.4 shows the distribution of errors in four hypothesised systems with differing precision and accuracy. These quantities have been used to survey and summarise the performance of numerous location systems [71]. The Minimum Performance Level (MPL) of the system specifies the accuracy and precision of a system with additional detail on contributing factors such as sensor density [72].

However, accuracy and precision are specifically suited to describing systems with Gaussian (Normal) distributions. Systems with more complex error distributions should not be expressed in this manner. The use of surveyed RF signal intensities in RADAR is likely to give rise to non-Gaussian distributed error because of the large (non-Gaussian distributed) changes in the received signal strength caused by small-scale fading.

An alternative approach which specifies a single probability for a particular sighting [89] is also not sufficient. This technique fails in particular for systems which resolve a number of equally likely positions. For example, the set of possible location estimates derived from knowledge of the distance to a single reference points forms a ring of equally likely locations.

(a) Single Transmitter Ranging (b) GPS (c) Active Badge

Figure 2.5: Location expressed as a Probability Density Function

Probability Density Functions (PDFs) provide a more expressive representation of uncertainty information [91, Chapter 6]. These functions encode the likelihood that a given object was located at a given position at the given time. Figure 2.5 shows three hypothesised uncertainty functions: (a) given a single estimated distance from a transmitter the locus of likely positions forms a ring around the transmitter; (b) dilution-of-precision in the GPS satellite constellation presents an ellipse of (two-dimensional) position uncertainty; (c) a sighting by a particular sensor in the Active Badge system may have arisen from a badge positioned in a neighbouring room. Specifying uncertainty as a PDF naturally expresses the combined probabilities of information from multiple sensor systems [7]. This representation could be extended to include all the dimensions of uncertainty such as time of sighting and the identity of the tracked object.

Uncertainty in location information also varies during the operation of the system [72]. A simple example of this is the Active Badge system which may occasionally report a sighting for adjacent rooms. The distribution of this error will change dependent on both which room the user is actually in and also where in the room the user is located.

The use of GPS for high accuracy surveying has prompted many studies regarding the sources of errors in the system. Sources of error include atmospheric delays, ephemeris (satellite position) deviations, clock drift and signal multipath [139, Chapter 4]. Atmospheric effects in the ionosphere form the dominant source of error [96]. These can be corrected by exploiting the signal diversity between the two GPS carriers (L1 and L2) [14]. Another major source of inaccuracy occurs when the receiver sees a reflected rather than direct path signal from the transmitter (multipath). Directional antennae with low gain near the horizon (most reflected signals arrive from low angles) alleviate this. Other approaches include antenna arrays (spatial diversity) and long-term observation of signals (temporal diversity). Given suitable environment models the multipath effect can be accurately simulated [28].

Understanding the error properties of location systems (other than GPS) has not been a priority for engineers. One explanation for this is that errors are traditionally only investigated for systems with demanding precision requirements. Sentient Computing is unusual in this respect because applications can often tolerate low precision data but will still require good estimates of error. Techniques for achieving high precision and accuracy are often resource intensive. In particular, the correction of errors in GPS measurement often relies on additional hardware or measurement time. The resource limited nature of Sentient Computing might limit the applicability of this approach.

2.3 Fault Tolerant Systems

One technique for vacuously achieving dependability is to provide a system which *never* produces data and doesn't claim to do anything different. The other extreme is more useful: to provide a system which never fails. The construction of Fault Tolerant Computer Systems (FTCS) has been the focus of research and commercial application for many years. Johnson provides a number of useful definitions [78]:

- **Faults** are defects or flaws within hardware or software components;
- **Errors** (caused by faults) are system states that are incorrect as compared with the system specification;
- **Failures** are instances of the system failing to perform correctly due to an error.

A system is regarded as *fault tolerant* if it is capable of continuing to function correctly in the face of faults. A specification of system behaviour must be given in order to define the system's correct actions.

The definition of dependability used in this dissertation differs to that of fault tolerance in the dynamic nature of the specification of system behaviour. A dependable system evolves its specified performance (e.g. accuracy) in light of current system status. It is acceptable for the system to stop producing valid information only if this fact is advertised to applications.

Johnson also gives terms for specifying aspects of a system's fault tolerance:

- **Reliability** is defined as the probability that the system performs continuously, without failure throughout a prescribed period, given that the system was performing correctly at the start of the interval.
- **Availability** is defined as the probability that the system will be operating correctly at a particular point in time. Highly available systems may experience frequent periods of being inoperable as long as each period is extremely short.
- **Safety** is the probability that the system will either perform within its specification or fail in a manner that is considered safe with respect to other systems or users. It is possible that a system may be considered safe but also unreliable; the statement of safety concerns the failure state rather than the likelihood of arriving at it.
- **Performability** determines the level of service that the system provides. It is the probability that the system will provide a particular level of service at a particular point in time.
- **Maintainability** specifies the probability of a system being restored within a particular period of time from the point of failure.
- **Testability** is related to maintainability in specifying the ability to test for particular attributes within a system.

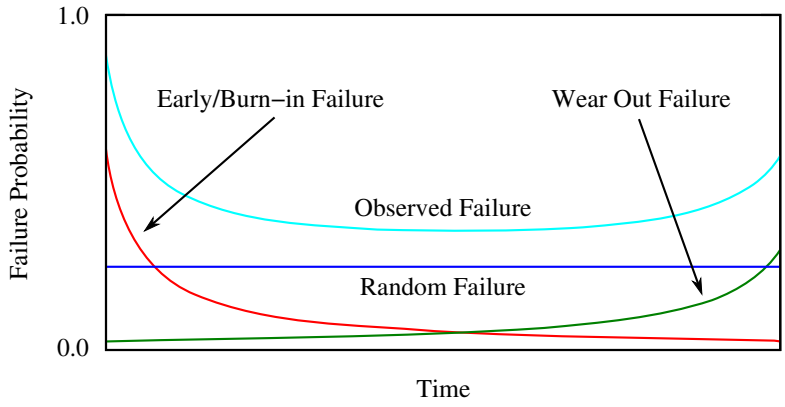


Figure 2.6: The *bathtub* curve for the probability of hardware failure

Quantities such as safety, maintainability, and testability remain directly important for dependable sentient systems. Programming concepts which are useful to sentient applications such as containment events are directly affected by the reliability of the system: a containment event relies on continuous system operation between two consecutive sightings of a user in order to ensure that the event is raised at the correct time. The availability of the system is important for information-polling sentient applications in order to ensure that a query can be answered at the required time instant. Performability must now encode the multiple levels of service possible from the system as it adapts to failure. Applications select a level of service with a suitably high performability for the current instant in time.

2.3.1 Hardware failure

One common method for specifying the reliability of a component is the Mean Time To Failure (MTTF) and the Mean Time Between Failures (MTBF). However, unless the failure probability of the device is exponentially distributed (a memoryless distribution) this value has little meaning. It is commonly the case that the failure probability over time forms a *bathtub* curve [42, pp. 108–121]. This curve is formed of the decreasing failure probability due to *burn in*, a constant random failure probability and an increasing failure probability due to *wear out*; shown in Figure 2.6. Interestingly the same curve is evident in plots of human mortality against age [56].

The effect of high early failure rates of devices can be mitigated by exercising (burning in) the device before it is integrated into the system. Similarly, the probability of failure due to wear-out should be included in the maintenance and replacement program for the system.

Most location sensing systems in Sentient Computing are based around solid state sensors and so have an operational lifetime greatly in excess of the system itself. However, mechanical sensing systems in particular may experience increased failure rates from wear out of components. Piezoelectric floor sensing, utilised in the Magic Carpet system [107], can withstand only a limited number of impressions before wearing out and is damaged by overly high pressures.

Software components of a system typically do not demonstrate bathtub curves for failure probability. This is mainly due to the fact that software commonly does not physically degrade over time and so there is no significant wear out probability. Factors such as software upgrades and changes to the operating environment make the curve complex.

2.3.2 Redundancy

Fault tolerance at the hardware level is often achieved by replication of the components i.e. the addition of redundant components. The decreasing cost and size of semi-conductor components often make this a particularly cost-effective method.

Passive hardware redundancy

Passive hardware redundancy aims to achieve fault tolerance without requiring any action by the system. This is referred to as fault masking and is commonly achieved using voting mechanisms to arbitrate between redundant computation blocks. Triple Modular Redundancy (TMR) and, more generally, n-Modular Redundancy (nMR) relies on a majority vote based on the output of replicated hardware to mask faults. The number of voting blocks within a system represents a trade-off between system throughput, hardware costs, and granularity of detection.

Active hardware redundancy

Active hardware redundancy provides infrastructure to detect faults and remove faulty hardware from the system. Active hardware redundant structures do not attempt to prevent faults entering the system; they instead aim to reconfigure the system and return it to a non-erroneous state within a specified time.

Typically, active hardware redundancy allows the system to use cold or warm spares. A cold spare must be installed into the system and brought online whereas a warm spare is kept in a more active state to minimise the activation time.

Error detection

Redundancy techniques rely on the system (implicitly or explicity) detecting an error in some component. These checks can take a number of forms [6]:

- **Replication checks** compare the result of identical functional units for consistency. n-Modular Redundancy is an example of this;
- **Timing checks** use an external clock source to validate that actions are occurring at the correct times or intervals;
- **Reversal checks** validate that the output is consistent with the inputs. This concept is examined in greater detail in Chapter 5;
- **Coding checks** utilise information redundancy to detect errors in the output. Cyclic Redundancy Check (CRC) codes can serve to detect errors in signals whereas Forward Error Correcting (FEC) codes have the capability to recover from a specified number of errors;
- **Reasonableness checks** confirm that the output conforms in some way to the specification of the system. One scheme [73] provides a set of checks which may be applied to system components. Examples include checking the monotonicity of a signal, or verifying that a signal falls within some specified bounds.

- **Structural checks** exploit features in the structure of the data to maintain consistency. Algorithm-based fault tolerance for matrix operations [76] is an example of this where additional checksum rows and columns are added to the matrix. The functional operations of the system are altered to preserve the checksum values.

Mobile devices play a key role in Sentient Computing but are often resource impoverished. Redundant components in a system are likely to increase both cost and power consumption and so designers might choose to exploit time- rather than space-redundancy by repeating calculations through the same component.

2.3.3 Fault tree analysis

Fault analysis seeks to build a model of system components. Propagation of component failure rates can be used to estimate the failure rate of the total system.

Fault tree analysis is used to analyse a system to determine possible sequences of events which can lead to a failure state [92]. Fault trees can include both hardware and software components in order to model the effect of combined errors from different areas of the system. Components are combined by way of logic gates which determine the conditions upon which a fault propagates through the tree. The fault tree represents the movement of faults, rather than data, through the system.

Classically, fault tree analysis considers only static fault trees constructed from AND, OR, and M-of-N type logic gates. An M-of-N gate propagates a fault if M of the N provided components fail.

The most immediate analysis of a fault tree is to derive the expected failure rate for the whole system. The emergent failure rate from an AND logic gate is the conjunction of the incoming failure rates, and so on. Fault trees are also used to derive *cut-sets* which consist of minimal sets of events which conspire to cause a failure. Cut-sets aid in achieving a target failure rate without over-engineering.

Fault tree construction is often disjoint from the conventional design process due to the need for specialist knowledge. Work to automatically synthesise fault trees from designs, expressed in specialist fault tolerance supporting languages [140] or Unified Modelling Language (UML) [106], aims to avoid this problem. Automatic synthesis will produce fault trees with more comprehensive system coverage limited only by the quality of the system design itself.

Error detection tests, such as reasonableness checks, can be placed with reference to a fault tree. This generally achieves better coverage than intuitive methods based on designers assumptions [74].

2.3.4 State dependent analysis

More sophisticated analyses require modelling of the state of the system. Dynamic fault trees extend the basic array of gates with stateful considerations such as:

- **Sequence Enforcing**. A simplifying constraint that the input events may only occur in left to right order. Outputs a logic 1 if all inputs are true;

- **Priority AND**. A failure is propagated only if all the inputs fail in the specified order.

Adaptive systems which degrade gracefully when faults occur might be modelled with each state representing a particular level of adaption.

In these cases analysis of the fault-tree is significantly more complicated. Common approaches utilise Markov analysis to derive the likelihood of the system being in a particular state [42, Chapter 6].

2.3.5 Software

Fault tolerance techniques can also be applied to software [69]. As software complexity increases so will the number of errors. Exhaustive testing also becomes infeasible and so software fault tolerance has an important role to play in compensating for programming and design errors.

N-version programming is a technique similar to nMR for hardware: a number of independently implemented programs provide results to an acceptance program which selects the correct result (often through voting). This approach is expensive because it requires independent teams of programmers working on the same problem. Different implementations of an algorithm cause further problems by requiring different amounts of CPU time to run to completion and so the throughput of the system will be limited to that of the slowest implementation. Also, care must be taken to avoid over-complication of the acceptance software which could also become a source of faults.

A recovery block system employs a *primary routine* to calculate the critical software function; an *acceptance test* to validate the result of the primary routine; and at least one *alternate routine* which implements the same function as the primary routine [113]. If the result from the primary routine fails the acceptance test then each alternate routine is used in turn until a result is obtained that meets the acceptance test. An exception is raised if every implementation fails to produce a valid result. Note that although the alternate routines implement the same function (to some level) as the primary routine they may be less efficient or capable. This system also requires a watchdog timer to ensure that routines do not loop indefinitely. Recovery blocks could be a possible architecture for implementing systems that degrade gracefully when faults occur.

Recovery Oriented Computing (ROC) [43] attempts to increase reliability by decreasing the time to repair/recovery. One technique used is Recursive restartability [29] which aims to increase system availability through "microrebooting" components. A graph of components is created with an edge between two nodes if the destination node requires notification if the source node is to be restarted. This aims to increase availability by minimising the impact of a restart to only the necessary components. One design point observed is that it is often possible for applications to trade precision/consistency for availability and so can better remain operational in the case of degraded operation caused, for example, by some parts of the system restarting. Successful application of this technique relies on detecting more than simply logical dependencies. The framework must know which instance of a particular component depends on another rather than simply that components of type A depend on components of type B.

One means for doing this is provided by PinPoint, a tool for automatically analysing the run-time operation of J2EE applications [30]. PinPoint constructs system dependencies (and detects failures) by monitoring message requests between clients.

Another ROC approach is that of reversible systems [25]. A reversible system allows the operator to rewind the current (presumably erroneous) system state and apply the required fix before replaying and returning to the current checkpoint. *External inconsistencies* are caused if any externally visible actions are taken erroneously by the system. These cannot be rolled back. Instead, the system must either decide if the errors can be tolerated by the user, or if corrective actions must be performed to explain or recover the external state.

2.4 Summary

Many Sentient Computing applications have been developed. Location information is a particularly useful form of context which is commonly exploited by applications. Other considerations such as the operating environment and the type of location information used vary between applications. Location systems have been built from a wide range of technologies each of which has different costs and benefits in terms of accuracy and precision, hardware complexity and coverage. Dependable systems should select location technologies based on the ability to accurately model their behaviour.

The task of a dependable system can be made easier through the application of techniques used in Fault Tolerant Computing Systems. However, many of the techniques involve the addition of extra hardware or software components. This limits their application to resource limited systems (common in Sentient Computing).

Chapter 3

A Platform For Investigating Dependability

The construction of dependable applications without the support of an infrastructure which supports dependability is unlikely to succeed. Consider, by analogy, the use of Secure-Sockets Layer (SSL) for securing Internet web-browsing. It might appear at the outset that a secure channel may be constructed on top of an insecure TCP/IP connection but additional factors detract from this design. Firstly, there is the problem of key distribution: some out-of-channel mechanism must be used to distribute trusted keys. Secondly, although SSL provides secrecy and integrity it cannot guarantee data delivery—a malicious node within the network can prevent all communication. To guard against issues of this nature which might arise, this investigation begins at the lowest level of the system. In the case of Sentient Computing this implies beginning with the location system.

Cantag is a generic framework for building machine vision location systems. It is designed to aid investigation into the dependability of location systems by permitting in-depth instrumentation of the recognition process and comparison of different processing algorithms. Cantag also embodies particular contributions for improving the reliability of machine vision systems through dependable data coding techniques, and robust pose extraction.

Parts of the implementation of Cantag have been provided by other members of the Digital Technology Group. Specifically, thanks are due to Robert Harle for the implementation of the camera distortion correction algorithms and the non-linear, back-projection routines for square tags. Thanks also, to Alastair Beresford for his work on Cantag's eigenvector solving routines and general image handling classes. The work on rotational invariance coding schemes arose from discussions with Christopher Cain from the Department of Pure Mathematics and Mathematical Statistics. He is responsible for the conception and implementation of the Structured Cyclic Coding (SCC) scheme.

3.1 Design Goals

Investigating, and supporting, dependability places a number of requirements on a system.

- **Reliable Implementation.** Any study of faults particular to Sentient Computing will require the rate of software and hardware faults to be significantly less than the rate of errors due to algorithmic or runtime problems. Numerous examples from software engineering show that this is no small requirement [147].

- **Ease of Instrumentation.** Once a failure has occurred it is important to identify the factors that contributed to it in order to incorporate mechanisms to cope. The system must be instrumentable to allow back-tracking to locate the causes of these failures. Common design techniques such as encapsulation mean that this kind of instrumentation is problematic to add to existing systems.

- **Flexibility.** Location system hardware and tracking implementations are complex and so it must be possible to tune and adjust the processing stages within the system to gain insight into limiting factors and critical steps.

Conventional requirements of a location system (such as coverage area) do not include particular emphasis on instrumentation or flexibility. This makes the use of existing location systems (either commercially available or research-based) problematic.

3.2 Machine Vision Systems

Due to their low cost, high reliability, and increasing ubiquity, video cameras and lenses are a good choice for hardware to underpin a test-system. Use of this commodity hardware running on a desktop PC will result in a system mostly implemented in software which permits easier instrumentation and tuning as well as reduced development costs as compared to a system requiring specialist or esoteric hardware. As described previously, machine vision systems already find use in a number of applications and are capable of producing fine-grained metric location information.

Cantag is a Marker-Based Vision (MBV) system which tracks two-dimensional fiducial markers deployed throughout the scene. The tagged nature of the system is exploited to provide a fast and reliable tracker. Fiducial markers can be thought of as advanced bar-codes (often printed using commodity printing hardware) with the potential not only to label an object but to position it accurately.

The operation of an MBV system can be viewed as the derivation of a transformation between a tag and camera coordinate frames. The *tag coordinate frame* is measured relative to the surface of the tag surface. The *camera coordinate frame* is determined relative to the viewing camera—as the tracked tag is moved its coordinates in the camera frame will change. The process of *back-projection* is used to determine the set of points in camera coordinates which correspond to a given point in the image [64, Chapter 6]. For systems composed of more than one camera it is important to provide location information without reference to the viewing camera. The *world coordinate frame* specifies positions relative to some global origin.

The field of Augmented Reality (AR) has been the traditional development domain for such Marker-Based Vision systems where they are favoured for their dependence only on commodity hardware (decreasing deployment costs) and for their high degree of precision and accuracy

across six degrees of freedom (ideal for image-object registration). Most AR applications focus on *video overlay* where three-dimensional models are rendered into the video stream viewed by the user. However, MBV systems are capable of producing a variety of outputs. Bar-code reading is the simplest form of output where the identifier information or data stored on the sighted tag(s) are returned. More sophisticated information about the position of the tag can also be derived for applications that require it.

- **Two-dimensional** information pertains to the position and dimensions of the tag in the image produced by the camera. Systems such as the SpotCode reader [124] and Rohs' mobile phone tag reader [119] utilise two-dimensional information from the tag image to decode the data on the tag and perform simple visual overlay.

- **Projective** information allows the projection of three-dimensional models onto the image. This is typically used by Augmented Reality applications for visual overlay. Projects such as Handheld Augmented Reality [141] perform video overlay on a handheld PDA requiring projective information from the image. However, due to the impoverished nature of the PDA platform a low accuracy processing pipeline may be appropriate in order to achieve low power consumption and high performance. The MagicBook [20] application overlays active content onto the pages of a book and therefore also requires projective information.

- **Three-dimensional** information refers to the position and pose of the tag in the camera coordinate frame. This information may be used for applications requiring spatial reasoning. The Free-D system [136] is used to track the position of a mobile camera on a TV set. Marker tags based on concentric circles are located and the position of the camera is derived from the rays running through the centre of each target. Additional utilisation of the three-dimensional information from the marker tags adds additional constraints which will improve the location accuracy of the system.

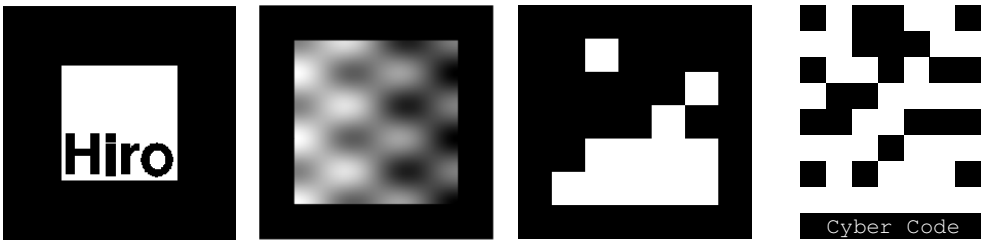
3.3 Fundamentals of Tag Design

The tag design used in an MBV system specifies the appearance of the family of tags tracked by the system. The algorithms used for recognition and tracking are not part of the tag design specification and may be changed by system engineers as needed throughout the lifetime of the system. Figure 3.1 and Figure 3.2 show a sample of the many unique tag designs in use. However, there are common aspects across all designs; these form the basis for the abstraction used in Cantag.

3.3.1 Data coding

The mechanism for encoding data onto marker tags determines whether a tag may hold general symbolic data (such as a URL) or only an identifier. The data coding scheme also affects the

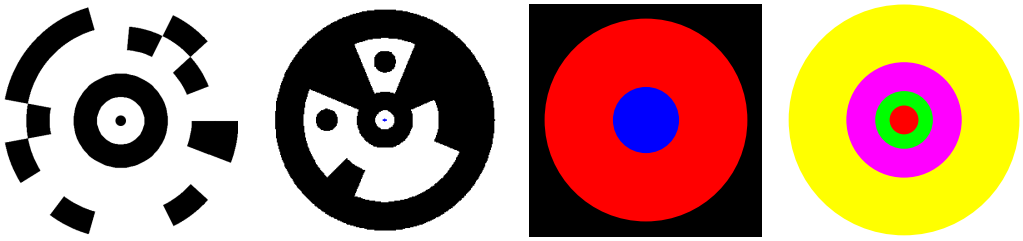
[1] `http://www.denso-wave.com/qrcode/index-e.html`



(a) ARToolKit [21] (b) Owen *et al.* [105] (c) Matrix [114] (d) CyberCode [115]

(e) QR code[1] (f) Rohs [119] (g) Zhong *et al.* [153]

Figure 3.1: MBV Systems using square tag designs



(a) TRIP [36] (b) VIS-Tracker [100] (c) State *et al.* [131] (d) Cho *et al.* [31]

Figure 3.2: MBV Systems using circular tag designs

susceptibility of the system to false negative results (failure to read a visible tag) and false positive results (detection of a not-present tag). Tag coding schemes broadly classify into *template* or *symbolic*. The typical mode of operation for template-based systems is to select the best match for the tag image from a database of issued tag images. Symbolic systems operate by applying a decoding technique to symbolic data sampled from points on the tag. Template-based systems benefit from the ability to attach semantic meaning to each tag for the user's benefit. Symbolic systems benefit from a larger, better defined, space of valid messages.

Figure 3.1(a) shows a tag from ARToolKit. This template coding system uses the estimated transformation from tag to camera coordinates to apply a perspective correction to the tag contents before comparing the resulting image using an auto-correlation function against the set of possible templates. The best match (above a fixed threshold) is returned [21]. There are numerous problems with the template technique. Firstly, the system designer must select a set of templates which are as distinct as possible. Owen *et al.* approach this problem through the systematic generation of template images by exploiting orthogonalities in the auto-correlation function [105]. However, as shown in Figure 3.1(b), any semantic meaning in the template design of the tag to users of the system is lost. The space of available templates is also smaller than a designer might imagine because tags are often rotationally symmetric and so each template must be distinct to all rotations of other issued templates. Another major problem with template-based tags is that currently no analysis is available of the effect that the distortion introduced by pixellation and perspective correction will have on the auto-correlation function.

Symbolic coding schemes divide the tag payload area into a number of datacells. The message is recovered by sampling the image at each datacell. This technique has a number of advantages over template coding methods: it is not necessary to linearly search through the set of issued codes to recognise a tag; the distinctness of two codewords is quantifiable as the Hamming distance; and the effect of perspective distortion can be modelled as bit-errors in the data.

Cantag currently only implements symbolic coding schemes but is easily extendible to incorporate template coding methods.

3.3.2 Tag shape

Fiducial markers are designed to provide projective invariants (properties which remain constant under projective transformation) which allow them to be recognised in the camera image. Designs commonly consist of a payload area which contains either a template or symbolic code and identifying features to aid location of the tag and recovery of payload data.

Square tags (shown in Figure 3.1) exploit the projective invariant that a straight line in three-dimensional space will also be a straight line in the resulting image. Thus, a square tag will transform into a quadrilateral in the image. The four corner points of the quadrilateral and the constraint that the tag has four-fold rotational symmetry provide sufficient information to perform the back-projection. It should be noted that the four corner points of a rectangular tag do not contain enough information to do this because the uneven edge lengths add additional unknowns to the system.

Cantag currently provides a single, generalised, square tag design. This design is parameterised on the number of datacells (n) along one edge of the tag. This gives a total data payload of the order of n^2 bits. The design divides the tag area into a grid of $n+2$ rows and columns. The two

edge rows and two edge columns are set to black to form the border of the tag and the remaining grid cells contain the symbolic data payload.

There is no reason why one cannot perform back-projection on regular n-sided polygons ($n > 4$). The corner points of these tags over-constrain the back-projection problem and so provide some tolerance to errors in the estimates of the corner vertices.

A circular tag may be considered the limit of an n-sided regular polygon. Figure 3.2 shows some existing circular tag designs. Circles possess the projective invariant that any perspective transform will result in an ellipse in the final image [44, pp. 256–261]. The imaged ellipse defines two possible orientations for the circular tag due to the fact that a circle in three-dimensions tilted towards the camera by θ degrees produces the same imaged ellipse as the circle tilted away from the camera by θ degrees (Figure 3.3). Additional information from the image is required to disambiguate the two possible back-projections. Possible sources are the inner edge of the bullseye or the position of the datacells.

The circular tag designs provided by Cantag divide the data payload area up into datacells over a fixed number of rings and sectors (Figure 3.4), each of the data rings is fixed to the same width. Another possible scheme might permit the division of each ring in to a different number of sectors—this would allow for fewer sectors on the inner ring where the datacells are small in size. Cantag allocates the same width to each data ring. Alternatively, the width of the data rings might be varied to provide a portion of the tag which can be read from an increased distance (larger datacells). The investigation of these possibilities is left to future work. There are three types of circular design available:

- **CircleInner** tags have the target bullseye contained in the centre of the tag inside the data rings. This maximises the area of the tag available for the data payload at the expense of reducing the size of the target bullseye. See Figure 3.5(a).
- **CircleOuter** tags have the target bullseye entirely outside the data area of the tag. This reduces the size of the data payload area but increases the size of the localising features of the tag. See Figure 3.5(b).
- **CircleSplit** tags overlay the data payload onto the target bullseye—the outer edge of the target bullseye is outside the data area of the tag and the inner edge of the target bullseye is inside the data area. This design maintains the large feature size of the outer bullseye edge and permits an increased data payload size than the CircleOuter tag. See Figure 3.5(c).

The circular designs are parameterised over six variables: number of rings, number of sectors and radii of bullseye-inner-edge (b_i), bullseye-outer-edge (b_o), data-area-inner-edge (d_i), and data-area-outer-edge (d_o). The relationship between the sizes of the bullseye and data areas determines the tag type:

CircleInner	$b_o < d_i$
CircleSplit	$b_i < d_i \wedge b_o > d_o$
CircleOuter	$b_i > d_o$

The circular designs featured in Figure 3.5 all have different relative widths allocated for the bullseye and data sections of the tag in order to optimise the payload storage. This technique is explained in Chapter 4.

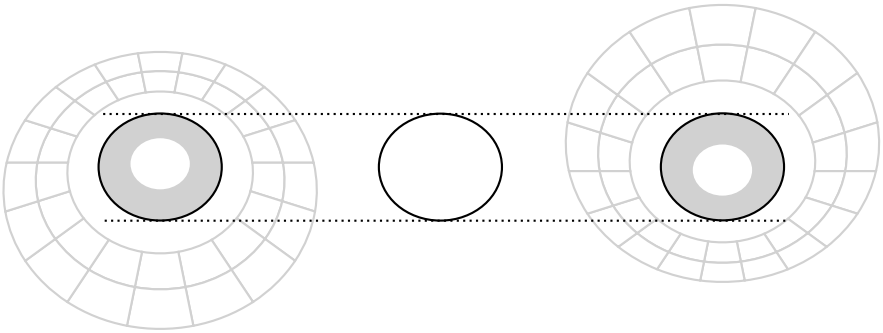


Figure 3.3: Ambiguous interpretations for the pose of a circular tag

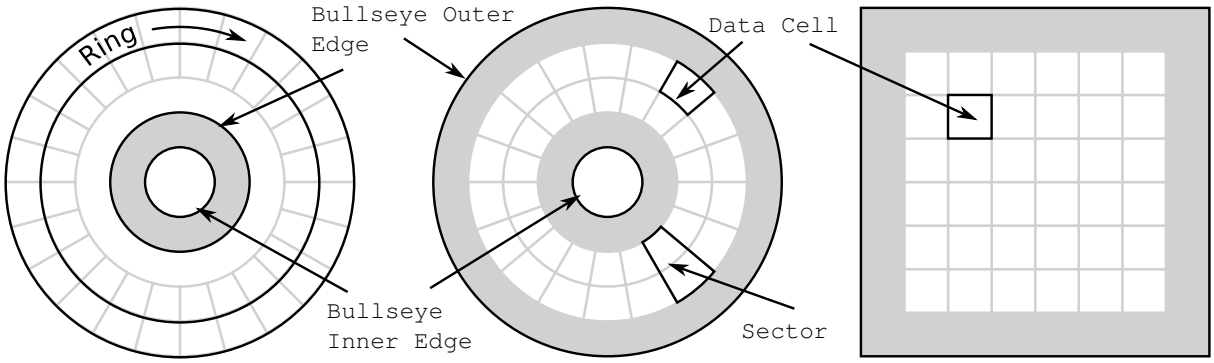
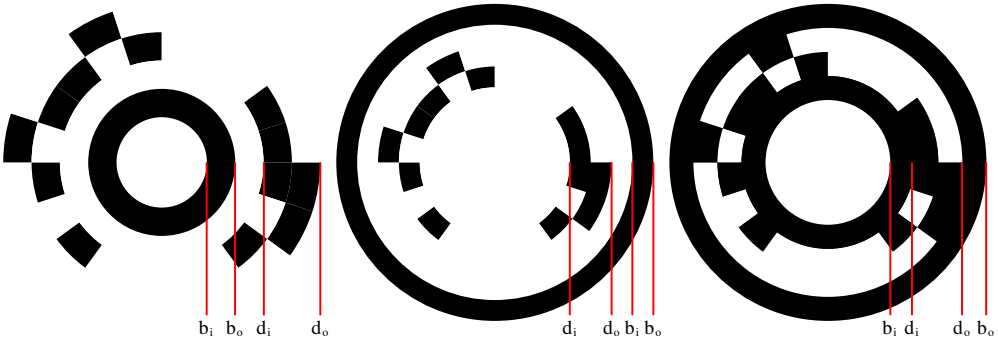


Figure 3.4: Tag design terminology



(a) CircleInner (b) CircleOuter (c) CircleSplit

Figure 3.5: The three generalised circular tag designs provided in Cantag

3.4 Image Processing Pipeline

Cantag can construct many different vision processing pipelines from the set of available algorithms. General operation of the vision pipelines can be summarised in the following steps:

1. **Image Acquisition**: The image for processing is captured from an image source;
2. **Thresholding**: The grey-scale (or colour) image is converted to a 1-bit image where *foreground* elements are white and *background* elements are black;
3. **Contour Following**: Contours are extracted from the thresholded image;
4. **Distortion Correction**: Lens distortion effects are removed from the extracted contours;
5. **Shape Fitting**: Basic shapes (quadrilaterals or ellipses) are fitted to the contours;
6. **Transformation**: The back-projection of the shape is derived;
7. **Sampling**: The encoded data on the tag are sampled from the thresholded image;
8. **Decode**: The sampled data are decoded according to the coding scheme used on the tag.

There are numerous other options available for the design of a MBV system: designers might choose to work using greyscale images and edge detection algorithms rather than thresholding and contour following. Similarly, the use of Hough transforms for fitting the edges of square and circular tags is another option. Cantag can accomodate these techniques with minimal additional effort to that of a standalone implementation of each algorithm. The intent here is to analyse one set of options which are used in today's existing systems and to demonstrate the approach one might apply to extend the analysis to other alternatives.

Cantag operates in a stateless fashion and tracks tags without reference to previously found locations. This is a useful starting point even for stateful trackers which incrementally track tags because it examines how the target tag is acquired in the first instance.

3.4.1 Entity abstraction

The basic data abstraction in Cantag is that of an *entity*. Entities correspond to distinct data items in the machine vision pipeline. For example, an initial image entity is subsequently broken down into a set of contour entities, the shape fitter converts a subset of the contour entities into shape entities, and so on.

Cantag consists of a set of *algorithms* that operate on entities. Each algorithm is implemented as a C++ function object which specifies the particular types of entity used as arguments or returned by the function. This approach allows each algorithm to be unit tested in isolation.

One of the design criteria for Cantag is that it should be possible to examine the intermediate stages of computation for any particular result. The *ComposedEntity* class is a generic programming technique developed to meet this need whilst maintaining loose-coupling between algorithms and without sacrificing performance. This is achieved by combining a set of entities

```
struct EOL {};

template<class H, class T = EOL> struct TypeList
{
  typedef H Head;
  typedef T Tail;
};
```

Figure 3.6: A simple implementation of type-lists in C++

```
template<class List> struct ComposedEntity :
  public List::Head,
  public ComposedEntity<List::Tail>
{
  ComposedEntity() :
    List::Head(), ComposedEntity<List::Tail>() {};
};

template<> struct ComposedEntity<EOL> {};
```

Figure 3.7: The ComposedEntity class

together into a single class containing the functionality of each of the entities within the set. The resulting class is recognised by the C++ type-system as an instance of each of the component entities. Thus, the ComposedEntity may then be passed to any algorithm expecting any of the entities in the original set.

The ComposedEntity implementation makes use of the concept of a *type-list* [5, Chapter 3]. Figure 3.6 shows a templated C++ class implementing a type-list. Given three existing classes `A`, `B`, and `C`, the user may then define a type-list as follows:

```
typedef TypeList<A,TypeList<B,TypeList<C> > > ABCList;
```

The ComposedEntity class (Figure 3.7) recursively inherits from the class at the head of the type-list and an additional ComposedEntity built from the tail of the type-list. The inheritance tree for `ComposedEntity<ABCList>` is shown in Figure 3.8.

Combining the basic entity classes by type composition has a number of useful features:

1. Algorithms may be implemented, compiled and tested without knowledge of any entity types other than those explicitly required by the algorithm itself.
2. The ComposedEntity class may be passed to algorithms from the client application in a type-safe manner—the client need not worry about extracting information from the ComposedEntity class to pass to the algorithm.
3. There is no need for pointer indirections, or virtual functions (also typically implemented through pointer indirection) when using the ComposedEntity class because the compiler

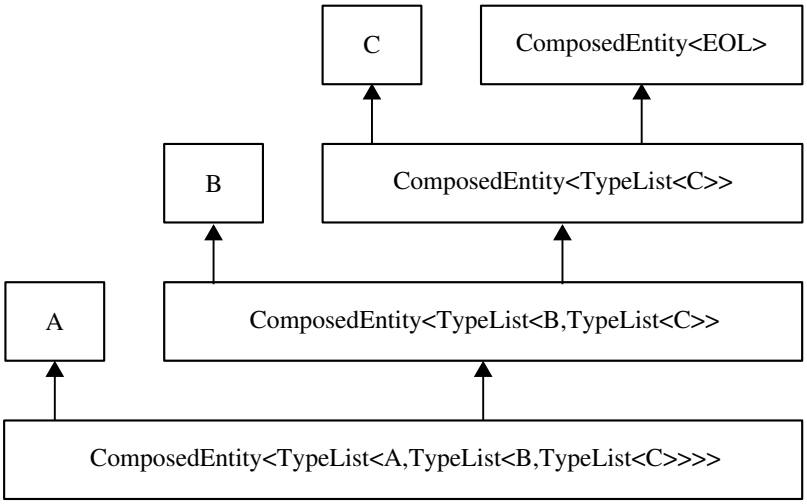


Figure 3.8: Inheritance diagram for an example ComposedEntity

can determine at compile-time exactly which parts of the ComposedEntity to pass to the various algorithms which are called; and

4. Much of the data that contributed to a particular result are stored in the same object allowing easy back-tracking when investigating problems.

3.4.2 Image acquisition

In addition to conventional image sources such as cameras and pre-recorded video, Cantag includes an image source for producing artificial images using OpenGL.

The OpenGL image source can be used as a test harness to render tags with arbitrary positions and poses. These tags are subsequently processed by the system, and the resulting data compared against the initial ground-truth input data. This mechanism provides a vital means to ensure that the algorithms offered by the system are correctly implemented. However, it also provides a means to understand the relative performance of different tags and algorithms since it allows huge numbers of images containing a variety of tag orientations to be systematically simulated.

The images produced by this test harness can be considered ideal: there is no camera distortion, lighting affects, or measurement error: the only source of error is derived from the pixellation of the image. Hence this harness can be used to place a quantitative *upper bound* on the capabilities of a specific tag. Thus, in addition to providing a means for comparing two possible configurations of the Cantag system, it can be used to answer questions as to whether some performance needs are actually possible with current algorithms.

3.4.3 Thresholding

The thresholding process converts grey-scale or colour images into 1-bit black-and-white images. Thresholding may be regarded as an image segmentation problem in which the image is

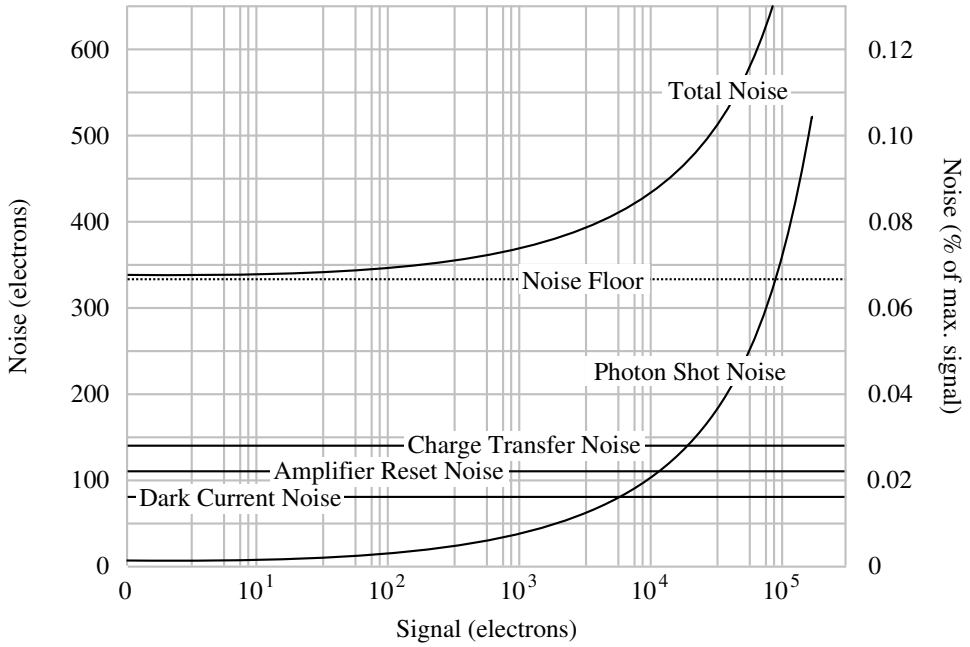


Figure 3.9: Noise sources in images captured from CCD arrays [127]

partitioned into foreground and background elements. The two major sources of error in the thresholding process are due to strong lighting variation and noise from the image acquisition phase.

Noise in the acquired image comes from a number of sources [127]. Examples include noise from *dark current* arising from thermal effects in a semiconductor (which is produced at a rate dependent upon the temperature of the sensor material). Other effects, such as *photon shot noise*, are dependent on the intensity of the received signal. Photon shot noise varies with the square-root of the signal level and arises from the statistical nature of sensing process which is dependent upon the random arrival and absorption of photons [127]. Figure 3.9 shows a summary of the sources of noise in CCD captured images (reproduced from the original publication). Images in low light levels have poor signal-to-noise ratios (SNR). When the general light intensity increases, the noise floor is eventually crossed and the SNR improves. The noise floor represents the sum of the illumination invariant noise sources. Figure 3.10 shows an image taken in low light levels which has been post processed (equalised) to spread the pixel intensities over the full representable range. The effects of poor SNR are clearly visible as noise across the whole image. The conventional operating environments for MBV systems are well-lit and so a good SNR is expected for the acquired images. However, from the thresholding view point it is the difference between the foreground and background pixel intensities for the region of interest that defines the effective signal strength. This can be much less than the signal range of the whole image and so image noise effects can still manifest themselves.

Lighting variation is common in most operating scenarios for computer vision systems. In particular, any space with natural sunlight will often experience intense illumination and strong shadowing. The effect of this is that a particular pixel intensity might correspond to a white (foreground) element in one part of the image whilst the same intensity might correspond to a black (background) element in another part of the image. An example of this is shown in Figure 3.11 (image by Edward H. Adelson[2]). The pixel intensities of squares A and B are iden-

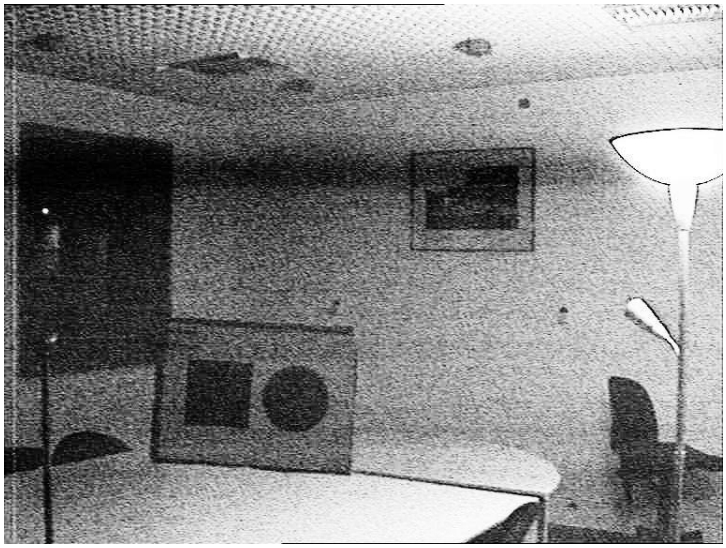


Figure 3.10: Photon shot noise in equalised low-illumination images

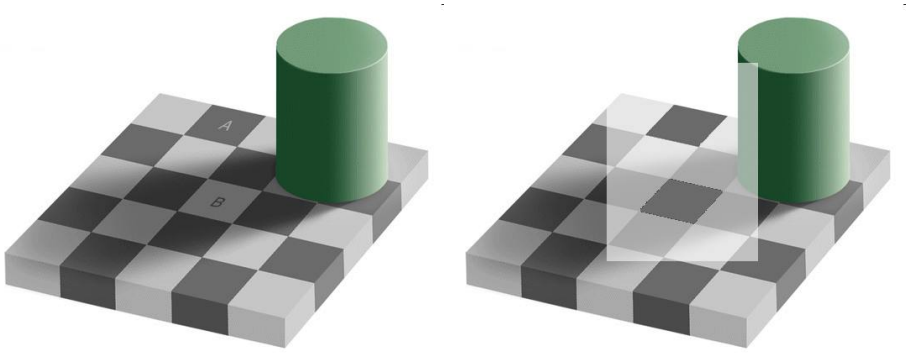
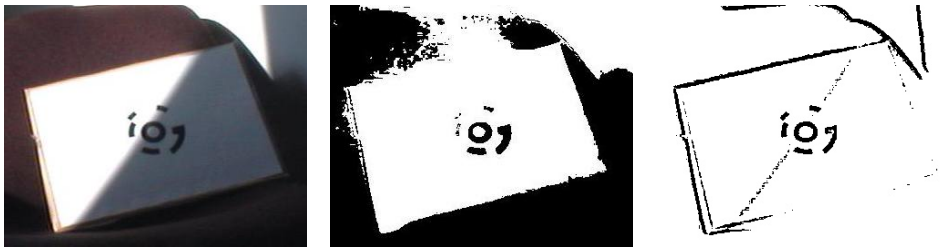
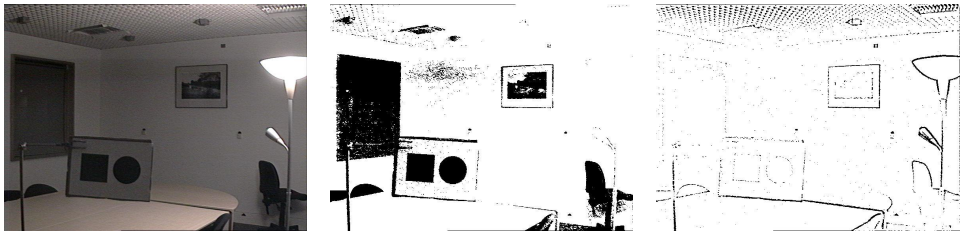


Figure 3.11: Optical illusion in colour interpretation: square “A” appears darker than square “B” despite them having identical pixel intensities



(a) Original Image (b) ThresholdGlobal (c) ThresholdAdaptive

Figure 3.12: Thresholding under varying lighting conditions



(a) Original Image (b) ThresholdGlobal (c) ThresholdAdaptive

Figure 3.13: Thresholding images taken in low illumination conditions

tical despite square A being darker than square B in reality. The right-hand image in the figure shows squares A and B with unchanged colouration. The surrounding pixels had their intensity reduced to highlight the similarity between the two squares. The images demonstrates how people viewing the image intuitively compensate for the lighting effects with additional real-world knowledge. Sophisticated (and computationally intensive) implementations of thresholding algorithms attempt to compensate for lighting interpretation problems by avoiding global (whole-image) analysis techniques and concentrating on local variations [137].

Cantag implements two thresholding algorithms: The ThresholdGlobal algorithm converts each pixel to either black or white based on whether it is larger or smaller than a fixed threshold. The ThresholdAdaptive algorithm maintains a moving average as it rasters over each row of the image. Each pixel is converted to black or white after comparison with the mean of the moving average from the current row and the previous row [150].

In situations where the illumination level varies strongly over the surface of the tag it is not possible to select a single threshold value to segment the image. Figure 3.12 shows an image of a tag partially illuminated by strong sunlight and the result of the ThresholdGlobal and ThresholdAdaptive algorithms. The ThresholdAdaptive algorithm clearly outperforms the ThresholdGlobal algorithm. The necessary parameters for both algorithms were set manually to give the best performance for this particular scene.

Conversely, there are also situations where the ThresholdGlobal algorithm performs better than the ThresholdAdaptive algorithm. An example of this is shown in Figure 3.13. In this situation the lighting variation does not detract from the performance of the GlobalThreshold algorithm because the variation is outside the region of the image containing the tag. The reason for the poor performance of the ThresholdAdaptive algorithm in this situation is because the variation

[2] `http://web.mit.edu/persci/people/adelson/`

between the foreground and background intensities in the tag region is small. If the sensitivity of the algorithm is increased to discriminate between them it becomes too sensitive to image noise.

3.4.4 Contour following

Cantag provides a topological contour following algorithm. The algorithm uses a border following technique [151] to build a tree of contours from the image [133]. The tree encodes the image topology—if the contour of interest wholly contains a second contour then the second contour will be a descendant of the first in the contour tree.

Execution of the algorithm proceeds by marking the original image with unique identifiers for contours as they are discovered and followed. The tree structure is then constructed as the contour follower moves across the image. When the raster position enters a contour the ID number is read from the image and set as the parent for subsequently discovered contours.

3.4.5 Camera distortion correction

Routine correction of the systematic distortions introduced by camera lenses has long been used in photogrammetry. Authors have long purported that it is unnecessary to attempt to build perfect lenses because the distortion introduced can be corrected to high accuracy [26].

The simplest camera model assumes no distortion and simply quantifies the transformation from camera coordinates to pixel values. This is based on the specification of the camera's intrinsic parameters: f_x, f_y specify the focal length/scale factor in the x and y directions, (u_0, v_0) specify the image centre (known as the principle point), and α specifies the skew—this has a value other than zero only if the pixels in the camera sensor array are not square in shape. For modern cameras f_x and f_y are often similar in value and α is not significantly greater than zero. This approach is suitable for low accuracy applications, especially if computation cost is a concern. The process of deriving these parameters is known as resectioning [64, Chapter 7].

The first improvement to this model is to incorporate radial distortion effects of the lens. Radial lens distortion causes the actual image point to be displaced radially, by an amount dependent upon the distance (r) between the image point and the principle point. The displacement distance is a polynomial in r whose coefficients are specific to the lens focus, and iris size. The model is further refined by incorporating "decentering" distortion which contains both a tangential and radial component [26]. Modern cameras and lenses often do not require a radial distortion polynomial of degree greater than four and often demonstrate minimal decentering distortion.

Derivation of the calibration parameters for a particular lens configuration may be achieved using commonly available camera calibration toolkits. The lenses used in this work have been calibrated using the Camera Calibration Toolbox for Matlab[3].

Cantag incorporates a number of possibilities for camera correction. The DistortionCorrectionNone algorithm performs a simple removal of the camera intrinsic parameters. This approach

[3] `http://www.vision.caltech.edu/bouguetj/calib_doc/`

is often suitable for two-dimensional bar-code reading MBV systems. The DistortionCorrectionSimple algorithm attempts to remove radial distortion using a direct (non-iterative) solution [94]. This method is computationally fast to apply and proves suitable for lenses with small levels of radial distortion. However, for large distortion effects (or for lenses with significant tangential distortion) the DistortionCorrectionIterative algorithm may be used to perform a non-linear numerical minimisation to precisely reverse the distortion affects. This approach is computationally intensive but provides the highest accuracy results.

3.4.6 Shape fitting

The shape fitting stage of the pipeline must fit either quadrilaterals or ellipses to the contours in the image as appropriate for the chosen tag design.

Ellipse fitting

An ellipse may be encoded either geometrically or algebraically. The geometric encoding of an ellipse consists of: a central point (x, y), the major axis length a, the minor axis length b, and the angle between the major axis and the horizontal axis θ. An ellipse may also be represented algebraically in the form of a generalised conic equation with parameters a to f:

$$ax^2 + bxy + cy^2 + dx + ey + f = 0 \qquad \text{where } b^2 - 4ac < 0 \tag{3.1}$$

Cantag performs conversion between these two representations as required based on existing techniques [45, Chapter 9]. Ellipse fitting is commonly performed using a least-squares approach where the algorithm attempts to minimise the maximum deviation of the contour from the ellipse. The FitEllipseLS algorithm in Cantag implements one approach published by Halíř and Flusser [60].

Another option for ellipse fitting is the FitEllipseSimple algorithm implemented in the SpotCode system. This algorithm derives the centre of the ellipse as the first central moment of the contour (the arithmetic mean of the coordinates of the edge pixels) and estimates the major and minor axes of the ellipse as the furthest and most proximate points on the contour from the centre. This algorithm is less computationally intensive and has a simpler implementation than the least-squares approach but is very sensitive to noise on the contour. Also, detecting the major and minor axes of the ellipse becomes an ill-conditioned problem as the ellipse tends towards a circle and so this algorithm tends to perform poorly for tags that are nearly fully facing the camera.

Quadrilateral fitting

Quadrilateral fitting requires an estimate of the four corners of the shape. The simplest approach to this is the FitQuadCorner algorithm which measures the local curvature of the contour and accepts the maxima as corners of the quadrilateral [134].

Another approach is the FitQuadPolygon algorithm which makes use of a line-simplification technique [41] to reduce the contour to a simplified polygon. Polygons of increasing degree

are hypothesised and checked against the original contour. If the maximum deviation from the contour is too large then an additional vertex is added to the polygon. Iteration ceases once the deviation is below a given threshold. If the resulting polygon has four vertices a shape-fit is deemed to have occurred.

The FitQuadConvexHull algorithm operates on the convex hull of the contour and partitions the points into corner-clusters by discarding points with low local curvature. If there are greater than four clusters then superfluous clusters are discarded in order of increasing edge length. The middle point of each cluster is then deemed to be a corner.

Fitting quadrilaterals in this manner is error prone because only four points on the contour contribute to the final result. Errors in the position of any of these four points cause direct errors in the final result. A more reliable technique is to allow all points on the contour to contribute to the final shape-fit. This reduces the impact of errors in position of particular points. The FitQuadRegression algorithm improves the estimates of the quadrilateral fitting algorithms. The original contour is partitioned into four edges based on the corners provided by the shape fitting algorithm. A linear regression is performed on each of the edges of the quadrilateral and the intersections of these four lines is computed as the new corner estimate. This produces behaviour similar to the FitEllipseLS algorithm whereby all points on the contour contribute to the shape estimate.

3.4.7 Transformation

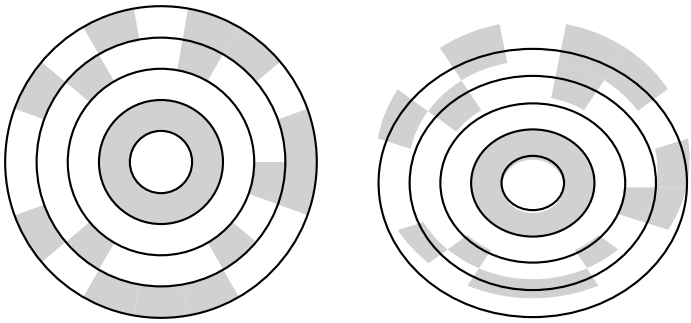
The transformation stage of the image pipeline derives a 4×4 homogenous transform matrix from the tag shape. This transformation converts the tag coordinate frame into the camera coordinate frame. Dividing the x and y coordinates in the camera frame by the z coordinates simulates the projection process and yields two-dimensional image coordinates. The derived transform information is used to estimate the points on the image to sample the binary code stored on the tag. Applications may also use the transform information for performing visual overlay or retrieving three-dimensional position and pose information.

Transforming ellipses

The simplest form of ellipse transformation is implemented in the TransformEllipseLinear algorithm which is based on the approach in the TRIP system. This approach estimates the positions of the tag's datacells by linear scaling of the fitted ellipse. Use of this algorithm within Cantag required that the linear scaling problem be cast into a transformation matrix. This can be achieved as follows based on the geometric parameters of the ellipse:

$$\begin{bmatrix} b\cos\theta & a\sin\theta & 0 & x \\ b\sin\theta & a\cos\theta & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transform information produced by this transform is only suitable for projecting back into the original image. The linear scaling is an approximation of the perspective effect and is



(a) Full facing tag (b) Tag Tilted at 30 degrees

Figure 3.14: Perspective effects on the TransformEllipseLinear algorithm

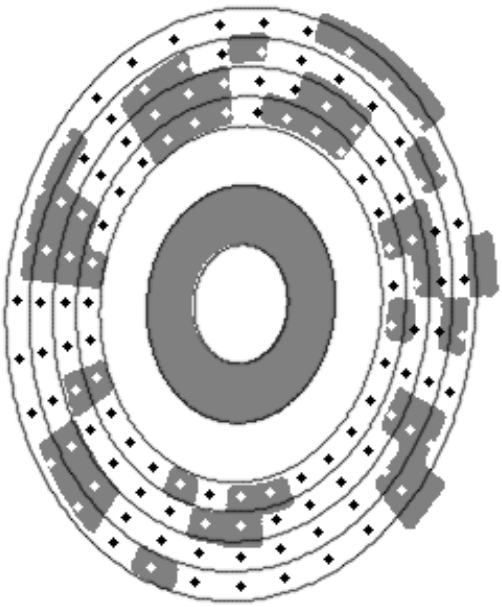


Figure 3.15: Large payloads are susceptible to perspective errors

based on the assumption that two concentric circles will, to a good approximation, become concentric ellipses in the image. This assumption introduces a systematic error in the projection called eccentricity error. Fully facing tags fit this approximation well but when there is a large distortion affect the approximation becomes less valid. This is shown in Figure 3.14. Ahn provides a formalism of this effect when measuring the central point of a circular tag [4] and finds that for a tag with small radius relative to its distance from the camera the error grows proportionally to the radius squared. This means that making the tag bigger does not improve the estimate of its central point when using this technique. For the current generation of circular tags with large datacells this effect has gone unnoticed by designers. However, for tags with a higher code density (and thus smaller datacells) this effect becomes noticeable. Figure 3.15 shows a section of a real image containing a circular tag tilted slightly away from the camera. Some error in the data point estimates occurs from other aspects of the system such as pixel truncation but errors attributable to the linear method are identifiable because the data points at

the right of the image are too close to the centre of the tag and the points at the left of the image are too far away from the centre of the tag.

The TransformEllipseFull algorithm is derived from Forsyth's "Pose from Circle" method [53] which was also used as a basis for the TRIP Adaptive Location System [35]. However, both of these algorithms contain subtle errors which are addressed later in this chapter (Section 3.6).

Transforming quadrilaterals

The simplest algorithm for transforming square tags is the TransformQuadProjective algorithm. The four corner coordinate pairs of a square (camera coordinates) are associated with the four corner coordinate pairs of the object. The camera coordinates are assumed to be an arbitrary linear combination of the object coordinates. The four corner points of the image quadrilateral, and their corresponding object coordinates, provide adequate constraints to solve for the camera coordinates [114]. This algorithm is highly susceptible to noise in the image because the constraints only enforce that the tag is rectangular, not that it is square. In the case of image noise this causes deformation of the derived object coordinate basis such that it is no longer orthonormal.

More sophisticated algorithms have been developed to better preserve the object coordinate frame. TransformQuadCyberCode is an implementation of the algorithm used in the CyberCode system [115]. This algorithm anchors a model of a tag at the centre of the imaged tag. The tag model is rotated about the anchor point until the discrepancy between inner angles of the model and the inner angles of the imaged tag is minimised.

Finally, the TransformQuadSpaceSearch algorithm implements a full non-linear search to minimise the difference between the projected corner points of a tag model and the imaged corner points [132]. The implementation of this algorithm requires careful selection of the starting point for the minimisation or the strong local minima for a tag with opposite inclination will interfere with the search for the true position. Although there is no fundamental geometric ambiguity, this problem is similar to the pose ambiguity problem for circular tags (Figure 3.3).

The techniques for deriving the back-projection of quadrilaterals are known as point-correspondence techniques: known points in the image are matched with points in tag coordinates and the resulting constraints are exploited to infer the three-dimensional position of the object. This approach is not possible for circular tags because given any point on the ellipse there is no way of telling which point on the circle (in tag coordinates) it maps to.

3.5 Dependable Coding

The current generation of symbolic tags do not take full advantage of the error handling potential of symbolic codes due to the rotational symmetry of the tags. Figure 3.2(a) shows a circular TRIP tag which contains two rings of data split into sectors. Each datacell in the sector can store a binary value and so each sector may store one of four possible symbols. One symbol (corresponding to a completely black sector) is reserved to orient the code in a *synchronisation sector*. The remaining sectors are used to encode two sectors of checksum information followed by the payload encoded as a base three number using the remaining three symbols. Despite

the (weak) error detection properties of the checksum the code is limited by the unprotected synchronisation sector. As a result this scheme can only ever guarantee to detect one bit of error; two bits of error suffice to fool the system into starting decoding from the wrong sector. Whether or not this invalid reading will pass the checksum depends on the data that was encoded.

Figure 3.1(c) shows a tag from the Matrix system which protects payloads using a CRC [114]. This approach lacks robustness because the tag has four-fold rotational symmetry. Thus, rotated tags read as permutations of the original code. No analysis has been presented as to the effect of these permutations on the Hamming distance of the code.

Figure 3.1(g) shows a square tag presented by Zhong et al. which carries five bits of data protected by a block sum code check or six bits of data protected by a Hamming code [153]. The four corner bits are used for orientation to ensure that the correct code can be read from the tag. Unfortunately, the block code does not protect these orientation bits and so two bits of error in the image can result in the system reading a rotated tag from the wrong orientation and thus returning an invalid code. The Hamming distance of this code should thus be considered to be only two bits until it can be proven that no two codes are rotationally self-similar.

Figure 3.2(d) shows a multi-ring circular tag design using *solid* rings chosen from a set of colours [32]. Assuming these colours can be reliably identified by the system, the method has the potential to be robust because the code can be read radially at any position. However, the amount of data that can be stored on the tag is small due to the large amount of redundancy. Also, additional coding would be necessary if error-correction capability was required.

3.5.1 Rotational invariance

Cyclic codes [59, Chapter 3] present a solution to the rotational symmetry problem. One property of a cyclic code is that any rotation of a valid codeword is a (different) valid codeword. If a tag's data are arranged in such a way that rotations of the tag correspond to rotations of the sampled data, rather than general permutations, then the error-detecting, or error-correcting, capabilities of the code will be unaffected by these rotations: suppose the minimum distance of the original code is d and given a valid codeword an error is introduced in fewer than d places. If the resulting word is equal to the rotation of some codeword then it itself must be a codeword (all rotations of codewords are codewords) and so there must be at least d differences between the two codewords. This is a contradiction, thus it is not possible to read a rotation of another valid codeword after undergoing fewer than d bit-errors and so the minimum distance of the code is unaffected.

Reading a code from a tag such that all symmetric rotations of the tag correspond to rotations of the code (termed *rotational invariance*) separates the coding scheme from tag design details and enables a mathematical analysis of code capability. It is possible to achieve rotational invariance for both square and circular tags.

Circular tags have rotational symmetry up to the number of sectors on the tag: a rotationally invariant ordering must yield only rotations of the codeword regardless of the starting sector. Figure 3.16 shows two orderings for reading data from a circular tag. The first ordering is not rotationally invariant whereas as the second ordering is. Consider, if the first reading of the tag reads cells in the order $1, 2, 3, 4, 5, 6, 7, 8, 9$ then the effect of a rotation by two sectors for the

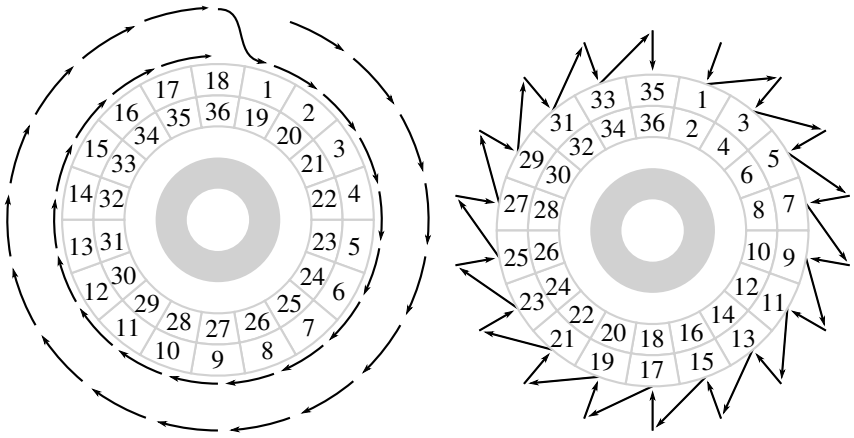


Figure 3.16: Reading a circular tag non-invariantly (left) or invariantly (right)

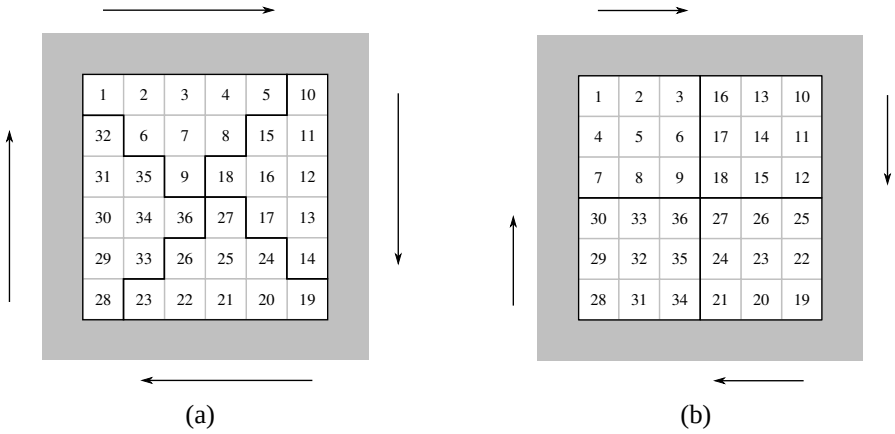


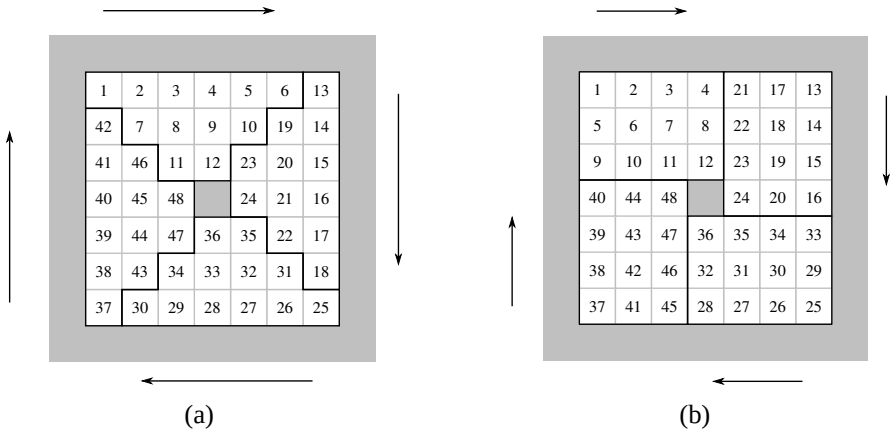
Figure 3.17: Reading a square tag in a rotationally invariant manner. A valid reading could start at any corner and progress in a clockwise direction

first ordering yields a permutation rather than a rotation of the original read order because a full 360 degrees of the outer ring are read from the start point before moving to the inner ring.

Square tags have four orders of rotational symmetry and so a rotationally invariant ordering must yield a rotation of the codeword for each starting quadrant. Figure 3.17 shows two rotationally invariant orderings for reading data from a square tag.

If a square tag has an odd number of datacells along one edge it is not possible to utilise the datacell at the centre of the tag. This is because each quadrant of the tag (corresponding to one unit of symmetric rotation) must have the same number of datacells in it. Figure 3.18 shows two possible orderings for tags with an odd number of datacells along one edge.

If a cyclic code is applied to a tag in a rotationally invariant manner then the error-detecting or error-correcting properties of the code will not be affected by the rotational symmetry of the tag. However, this presents an additional problem because the system will be unable to select the correct code from the set of possibilities read from the tag. Each possibility will appear as a valid codeword (after applying any applicable error correction routine). One approach is to select the particular rotation which has a smaller value than every other possibility. This means



(a) (b)

Figure 3.18: The central datacell is unused if the tag has an odd number of cells

that for each value coded onto a tag there will be a number of additional codewords which also decode to the same value. Codes exhibiting this property are termed *symbolic identifier* codes—the code cannot store arbitrary data.

3.5.2 Tag coding abstraction

The rotational symmetry of the tag is considered with respect to fixed features of the tag design. For a square tag the edges of the border determine four orders of rotational symmetry. For a circular tag the edges of the sectors determine the number of orders of rotational symmetry present. A tag rotation by one *place* refers to as the minimum rotation required to align the fixed features of the tag such that they are indiscernible from those in the original orientation.

The tag's data-carrying capability may be characterised in terms of two variables: **Symbol Size** is the number of bits allocated to storing each symbol. If the tag is rotated by one place and the code re-sampled, the new value should be identical to the previous value after a rotation through symbol size bits; **Payload Size** is number of symbols the tag can store.

The payload of a circular tag with m rings and n sectors undergoes a rotation of m bits when rotated by one place and so has a symbol size of m bits. Since each sector stores a complete symbol the number of symbols stored is n.

A square tag with an even number of cells along each side has $2p \times 2p$ cells (where p is an arbitrary natural number). A rotation of one place rotates the payload by a quarter of the total size: this yields a symbol size of p^2 bits. The payload size is four of these symbols. A square tag with an odd number of cells along the edge ($2(p+1) \times 2(p+1)$ total cells) cannot store data in the centre cell (Figure 3.18). This leaves a symbol size of $p(p+1)$ bits and a payload size of four symbols.

Coding schemes may be parameterised in a similar way. The number of different symbols corresponds to the size of the field used to define the polynomials in the cyclic code. For example, the various generator polynomials for a CRC are defined over the field with two elements (symbol size is one bit). Reed-Solomon codes, which are used for error correction on CDs and DVDs, can be defined for fields of size 256 (symbol size is eight bits).

If the tag symbol size is eight bits it can also store codes with a symbol size of four, two or one bits by packing multiple symbols into each rotational block. In general a tag with symbol size s_t can store a code with symbol size s_c if s_c is a factor of s_t.

The entire codeword must be stored on the tag or else a rotation is not guaranteed to produce another valid codeword. For cyclic codes the size of all codewords is given by the block length. This precludes the use of CRCs on most tags: the generator polynomial for CRC-CCITT (a sixteen bit CRC) has a block length of 32767 bits. Typically, a CRC is used with much smaller messages than this—the unused bits are assumed to be zero and not transmitted. This is not permissible for rotational tags. A circular tag carrying CRC-CCITT data would need 151 rings and 217 sectors. A selection of more plausible coding schemes are presented next.

3.5.3 Coding schemes

The tag coding abstraction frees implementors of coding schemes from the details of tag design: the shape of the tag and layout of the datacells are unimportant. The pertinant consideration for designers is whether the symbol and payload sizes of the tag and the coding scheme are compatible. A tag with a symbol size of s_t and a payload size of p_t is compatible with a coding scheme with symbol size s_c and a payload size of p_c if:

- $s_c | s_t$ (the coding symbol size must be a factor of the tag symbol size); and
- $s_c p_c = s_t p_t$.

To demonstrate the flexibility of this technique a number of coding schemes have been developed and tested within Cantag.

Simple Parity Code

A bit string with n parity bits at the end fulfils the criteria for a rotationally invariant code: every rotation of the coded data will also have valid parity. The code symbol size (s_c) is one bit (thus making it applicable to all tags), all payload sizes ($p_c > n$) are applicable. A codeword with n bits of parity is generated from $p_c - n$ data bits, n parity symbols are appended to the codeword to give the required parity. This is an example of a code that can only encode an identifier because the decoded message must be rotated round until the minimal value is found. However, even 1-bit parity achieves the same minimum hamming distance as the TRIP code and the "hamming code scheme" by Zhong et al. and can store considerably more data.

Independent Chunk Code

Given a tag with a large symbol size, each symbol may be considered as a separate codeword which is protected by an error-detecting or error-correcting sub-code. The first bit of each symbol is used to anchor the code: the first bit of the first symbol is set and the first bit of every other symbol is unset. For example, a square tag of size 8×8 has a symbol size of 16 bits and a payload size of 4 symbols. A 44-bit payload can be encoded in four 11-bit chunks. Each

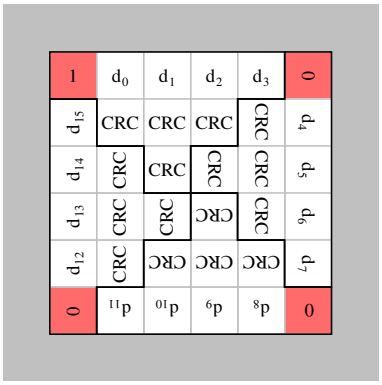


Figure 3.19: The Independent Chunk Code applied to a tag with large symbols, each of which contains an orientation bit and some error protection

symbol on the tag contains one chunk, one orientation bit, and a 4-bit CRC sub-code to detect errors in the chunk of data or in the orientation bit (Figure 3.19). This code is at least as strong as the 4-bit CRC used for each symbol; if the designer requires stronger error detection then a different sub-code can be used for each symbol. In the cases where errors occur evenly over the tag rather than concentrated in one sector this code should be rather stronger than a single 4-bit CRC. The drawbacks of using this code is that four bits of every symbol are used to get the same Hamming distance as traditional use of a single 4-bit CRC. Additionally a further bit is required per symbol to orient the code. The advantage of this encoding method is that the code need not have rotational invariance and so a truncated CRC is permissible.

The implementation of this code in Cantag is parameterised by symbol size, payload size, and the size of the CRC sub-code. The CRC polynomial is selected automatically at compile-time to maximise the Hamming distance of the resulting code [83].

Basic Cyclic Code

Conventional cyclic codes with codeword length equal to the payload size of the tag may be immediately applied to marker tags using rotational invariance. Due to the fact that all rotations of a codeword are also valid codewords there will be p valid interpretations for the value stored on the tag. The system deterministically selects one of these by returning the rotation of the codeword with the lowest numeric value. This means that the tag cannot store arbitrary data because only one in p messages can be recovered correctly from the tag. This is an example of a symbolic identifier code. It is only possible to store a unique identifier on each tag rather than storing arbitrary data.

Structured Cyclic Code

The Structured Cyclic Code (SCC) is a more conventional cyclic code with additional structure that encodes the amount of rotation that the code has undergone. The SCC produces codewords which encode the degree of rotation they have undergone. Consider, as an example, the string 0123456789, a rotation of four digits produces 4567890123. It is then trivial to recover the

rotation that the string (and thus the original message) has undergone by inspecting the number at the head of the string. SCC codewords efficiently combine this rotation information and the message data. For full details on SCC the reader is invited to consult the original publication [116].

Generalised TRIP Code

The coding scheme used in the TRIP system can be adapted for the rotational invariance abstraction. The original coding scheme supports tags with a *symbol size* of two bits and a *payload size* greater than three symbols—one symbol is reserved for the synchronization sector and two for the checksum. The generalised version of the code applies to any *symbol size* ($s > 1$) and applies an n symbol checksum onto any *payload size* ($p > n + 1$). The message to encode is treated as a number and encoded with base $2^s - 1$ onto $p - n - 1$ sectors. The checksum is computed by summing the value of each encoded sector and storing the result (with base $2^s - 1$) in the n sectors reserved for the checksum. The remaining sector is filled with the symbol $2^n - 1$—this is guaranteed to be unique and thus suitable for a synchronization sector because the remaining data was encoded with base $2^n - 1$ and so has maximum value $2^n - 2$.

3.5.4 Asymmetric tags

Another approach to solving the robustness problems caused by rotational symmetry is to introduce an asymmetric feature into the tag design thus permitting use of conventional coding systems. For example, as shown in Figure 3.2(b), the VIS-Tracker uses an off-centre eyelet for this purpose.

Introducing asymmetric features to a tag design is problematic. For reliable operation the designer must ensure that the asymmetric feature is strong (large) enough that the tag's orientation is never estimated incorrectly unless there are so many errors in the image that the chosen symbolic coding scheme would also fail. However, the designer must also minimise the size of any asymmetric features in order to maximise the amount of payload space on the tag. Any change in the coding scheme will also require re-evaluation of the size of the asymmetric features.

QR Codes are a popular two-dimensional bar-code that use a particular pattern on three corners to orient the tag; an example is shown in Figure 3.1(e). Four different levels of error correction are available of which level 'M' corresponds most closely to the level afforded by the SCC-2 code evaluated in the next section. QR Codes are available in a number of sizes, the largest of which has a data area with dimensions 177×177 bits. Some parts of the payload area are reserved for the three orientation patterns. This size of tag can store 18648 bits[4] which corresponds to a utilisation of 59.5%. An instance of the SCC code based on a Reed-Solomon code giving eleven symbols of separation between codewords applied to a circular tag with data area of 155 bits has a utilisation of 64%. The primary reason for this is that the area occupied by the asymmetric features added to the QR Code is disproportionately large compared with the error-correction capability of the error-correcting code.

Use of symmetric tags and rotationally invariant codes is advantageous in this respect because the minimum amount of payload space is wasted in order to encode rotation information. Also,

[4] See `http://www.denso-wave.com/qrcode/vertable4-e.html`

Name	Message Length (bits)	Hamming distance
TRIP	139	2 symbols
SPC	154	2 bits
ICC	93	2 bits per symbol
SCC-1	141	3 symbols
SCC-2	101	11 symbols

Figure 3.20: Data-carrying capabilities of the evaluated coding schemes. 155 datacells are available on the tag

rotationally invariant tags result in the least possible complication of the computer vision aspect of the decoding. The system need only read the data from the tag rather than search for additional features before decoding the information.

3.5.5 Evaluation

The OpenGL test harness was used to evaluate the performance of the new cyclic coding schemes. A circular tag with 5 rings and 31 sectors was used to carry a payload encoded with each of the proposed rotationally invariant schemes.

- **TRIP** Adaption of the original coding technique used in the TRIP system: 1 synchronisation sector followed by 2 checksum sectors and 28 payload sectors encoded base 31.
- **SPC** Simple Parity Code: 154 payload cells (not sectors) followed by 1 parity cell encoded base 2.
- **ICC** Independent Chunk Code: 31 independent chunks (one per symbol) containing 1 orientation bit, 1 parity bit and 3 bits of payload;
- **SCC-1** Structured Cyclic Code with f chosen as in a Reed-Solomon code giving 3 symbols of separation between codewords.
- **SCC-2** Structured Cyclic Code with f chosen as in a Reed-Solomon code giving 11 symbols of separation between codewords.

The data-carrying capabilities of each of these codes are given in Figure 3.20.

The OpenGL test harness was used to render fully facing tags at a distance of two tag widths from the camera. Gaussian noise was injected into the images and the target tags decoded using the full image processing pipeline. The three possible results from each test run are defined as:

- **Successful Read**: the payload on the tag is decoded and the returned code matches the value encoded (a true positive).
- **Failed Read**: the payload on the tag fails to decode and so the system fails to recognise a tag (a false negative).

Name	Successful Read		Failed Read		False Read	
	%	Normalised	%	Normalised	%	Normalised
TRIP	24.0	21.5	74.0	66.4	2.0	1.8
SPC	31.0	30.8	46.0	45.7	23.0	22.9
ICC	27.1	16.3	72.6	43.6	0.3	0.2
SCC-1	55.7	50.7	6.20	5.6	38.1	34.7
SCC-2	94.0	61.3	6.00	3.9	<0.1	0

Figure 3.21: Error rates for the evaluated circular tag designs

- **False Read**: the payload on the tag is decoded but the returned code does not match the encoded value, i.e. the error detection built into the code is defeated (a false positive).

Figure 3.21 shows the percentage of frames from 1000 trials for each code that contained successful readings, failed readings and false readings. Normalised values for these percentages are obtained by multiplying by the proportion of the utilised address space. The normalised value shows how efficiently a coding scheme copes deals with errors. Schemes which correct a lot of errors at the cost of a large reduction is payload will be penalised compared to schemes which cause a smaller reduction in payload.

The results confirm that allocating more bits to error control strengthens the code. The SCC-2 shows a particularly high successful read rate due to its large error-correcting capability. This redundancy also gives it a false read rate small enough that it failed to manifest itself in the 1000 samples. The error-correction ability of the SCC-1 code increases the successful read rate above that of the non-correcting codes at the expense of increasing the false read rate. The TRIP, SPC, and ICC codes have the same minimum hamming distance. However, the noise was evenly distributed across the whole image and so the ICC code's parity bits acted mostly independently giving it good false read rate. The TRIP code distributes the code particularly unevenly over the tag; this manifests itself in the code's more variable behaviour than the ICC code—it shows an increased successful read rate *and* an increased false read rate even though there is no attempted correction of errors.

A further experiment using the TRIP, SPC, and ICC tests was performed using a square tag of size 12×12 rather than a circular tag. The original description of the SCC code requires that the payload size is a prime number. This precludes their use on square tags for which the payload size can only be some multiple of four. Further refinement of the scheme has avoided this restriction [116] at the expense of increased implementation difficulty, computational cost, and space overheads in the coding scheme. The increased symbol size of the square tag means that the ICC code is a better choice than the extended SCC. Figure 3.22 shows the various decoding rates for square tags which bear out the same trends as for the circular tag. This provides some justification for the viability of performing code selection in isolation from the actual tag design. The ICC(square) code presents a better normalised successful read rate than the TRIP(square) code which is contrary to the results for circular tags. This is because the ICC code is much more efficient for tags with large symbol sizes and so its success rate is boosted to acknowledge this. However, the increased symbol size means that there will be fewer parity bits embedded in the code—this is reflected by the increased false read rate for ICC(square) over ICC(circle).

Name	Successful Read		Failed Read		False Read	
	%	Normalised	%	Normalised	%	Normalised
TRIP(square)	12.6	9.5	83.4	63.1	4.0	3.0
SPC(square)	27.4	27.4	44.0	43.7	28.6	28.4
ICC(square)	24.0	22.7	70.5	66.6	5.5	5.2

Figure 3.22: Error rates for the evaluated square tag designs

The choice of coding scheme is also affected by the usage scenarios of the final system. Scenarios with low image noise enable high utilisation codes to be used with little error detection capability. For interactive systems, designers might choose to minimise the false read rate at the expense of a higher failed read rate because users can be expected to retry a tag if it fails to read.

3.6 Back-Projection for Circular Tags

Existing algorithms for extracting three-dimensional information from circular tags include the "Pose from Circle" method [53] and its subsequent adaption for the TRIP Adaptive Location system [35], and Kanatani's method [80]. These algorithms derive information such as the normal vector and location of the tag.

The TransformEllipseFull algorithm improves on the "Pose from Circle" method in two important ways: firstly, the results of the algorithm are unified into a homogenous transformation matrix which expresses the full transformation from object coordinates to camera coordinates; secondly, a number of mathematical ambiguities which exist in the original technique that permit a number of geometrically impossible solutions to be derived are resolved.

3.6.1 Algorithm description

The back-projection algorithm derives the transformation from object (tag) coordinates to camera coordinates. This transformation is derived from the ellipse formed by the projection of the tag into image coordinates. The algorithm decomposes the transformation into a sequence of more simple steps. Each of these transformations is applied to the viewing camera thereby altering the projected image of the tag. After the final transformation the viewing camera coordinate frame is aligned with that of the tag and the projected image of the tag is that of a unit circle.

The first transformation rotates the camera such that the ellipse is aligned with the axes of the coordinate frame. From this position a second rotation is derived for the camera around the coordinate axis aligned with the major axis of the ellipse. The second rotation is chosen such that the projected image of the tag in this orientation is a circle. Finally, a translation and a scaling is applied to transform the projected image into a unit circle centred in the centre of the image.

The perspective transform is modelled by first drawing rays from the origin in camera coordinates ($\mathbf{c} = (x_c, y_c, z_c)$) to each point on the target circle. The projected image of the three-dimensional world is then formed by taking the intersections of these rays with the plane of projection which, without loss of generality, is assumed to be the plane $z_c = 1$. The image of the circular tag is then a slice through the cone of rays running from the origin to the target. The physical constraints of the camera prevent this slice from intersecting the base of the cone and so it is guaranteed that the image of the circular tag will be an ellipse.

The equation of this cone is a simple extension of the original ellipse equation (Equation 3.1). Notice that when $z_c = 1$ (the perspective plane) the equation reduces to that of the original ellipse:

$$\begin{aligned} ax_c^2 + bx_cy_c + cy_c^2 + dx_cz_c + ey_cz_c + fz_c^2 &= 0 \\ \begin{bmatrix} x_c & y_c & z_c \end{bmatrix} \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} &= \mathbf{c}^\mathsf{T}\mathbf{C}\mathbf{c} = 0 \end{aligned}$$

The process aims to find a sequence of transformations to apply to the perspective plane so that the slice through the cone is a circle. The equation is first modified to incorporate an orthogonal transformation $\mathbf{R}_1$ consisting of the normalised eigenvectors of $\mathbf{C}$ arranged in rows:

$$\mathbf{R}_1 = \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{e}_3 \end{bmatrix}$$

$\mathbf{R}_1$ is an orthogonal matrix and so $\mathbf{R}_1^\mathsf{T} = \mathbf{R}_1^{-1}$:

$$\mathbf{c}^\mathsf{T}(\mathbf{R}_1^{-1}\mathbf{R}_1)\mathbf{C}(\mathbf{R}_1^{-1}\mathbf{R}_1)\mathbf{c} = (\mathbf{R}_1\mathbf{c})^\mathsf{T}\mathbf{R}_1\mathbf{C}\mathbf{R}_1^\mathsf{T}(\mathbf{R}_1\mathbf{c}) = \mathbf{c}_R^\mathsf{T}\mathbf{C}_R\mathbf{c}_R = 0$$

Matrix $\mathbf{C}_R$ is the *diagonalisation* of $\mathbf{C}$—the leading diagonal consists of the eigenvalues of $\mathbf{C}$ and all other elements are zero. Applying $\mathbf{R}_1$ to the coordinate frame yields a second coordinate frame ($\mathbf{c}_R$), in which the equation now determines an ellipse with its major and minor axes aligned with the new x- and y- axes and the new z-axis running through its central point. This is evident because the diagonal matrix $\mathbf{C}_R$ will only provide non-zero coefficients for terms of a single variable in the ellipse equation.

A simple rotation of this coordinate frame parallel to the major axis of the ellipse will yield the original circle. Note that a rotation parallel to the minor axis will also yield a circle. However, this circle is not a possible geometric source of the projected image. The projection of a circular tag as it is tilted away from the camera is that the image of the circle remains the same width along the axis of rotation and is shortened along the perpendicular axis.

The original ellipse equation is only defined up to a scale factor and the signature (the balance of positive and negative eigenvalues) of the matrix $\mathbf{C}$ will be (2,1) [80] i.e. one eigenvalue will have the opposite sign to the other two. This allows normalisation of the matrix $\mathbf{C}$ so that it is has one negative eigenvalue and two positive eigenvalues. The eigenvalues are now ordered such that $\lambda_1 \geq \lambda_2 > 0 > \lambda_3$. This guarantees that the major axis of the new axis-aligned ellipse

will lie along the y-axis. This is because $1/\lambda_1$ and $1/\lambda_2$ correspond the lengths of x-axis and y-axis respectively. The largest *magnitude* corresponds to the major axis.

Arranging that the first eigenvalue is larger than the second has ensured that the major axis of the ellipse is aligned with the new y-axis in the coordinate frame. Consider a circular tag which is rotated around the horizontal axis. The resulting imaged ellipse will have its major axis along the horizontal and its minor axis along the vertical. Thus, the coordinate frame rotation required to go from the axis-aligned ellipse to the circular tag must be about the major axis of the axis-aligned ellipse. The original method proposed by Forsyth and subsequently used in the TRIP Adaptive Location system required that the eigenvalues are in increasing order but neglected to normalise the matrix $\mathbf{C}$. The intention was to align the major axis of the ellipse with the y-axis. However, if one of the eigenvalues happened to be a large negative number then the major axis was aligned with the x-axis instead—this resulted in a rotation of the axis-aligned ellipse about its minor axis which is not a valid model of the perspective effect. Another possibility [80] correctly deals with the sign of the eigenvalues but it subsequently proves impossible to calculate the angle of rotation required because the derived equations have non-real-valued solutions.

The previous step has simplified the ellipse equation to the state where a simple rotation ($\mathbf{R}_2$) about the y-axis will provide another coordinate system ($\mathbf{c}_{RR}$) where the equation is that of a circle, i.e. the coefficients of the $x_{c_{RR}}$ and $y_{c_{RR}}$ terms will be equal.

$$
\begin{aligned}
(\mathbf{R}_2\mathbf{c}_R)^\mathsf{T}\mathbf{R}_2\mathbf{C}_R\mathbf{R}_2^\mathsf{T}(\mathbf{R}_2\mathbf{c}_R) &= 0 \\
\mathbf{c}_{RR}^\mathsf{T}\begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}\begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix}\mathbf{c}_{RR} &= 0
\end{aligned}
$$

This equation specifies an ellipse. However, by suitable choice of θ it can be arranged that the coefficients for $x_{c_{RR}}$ and $y_{c_{RR}}$ terms are equal, yielding a circle rather than an ellipse. Expanding the above equation, setting the x and y terms equal, and solving for θ yields:

$$\theta = \pm tan^{-1}\sqrt{\frac{\lambda_2 - \lambda_1}{\lambda_3 - \lambda_2}}$$

The original requirement that $\lambda_1 \geq \lambda_2 > 0 > \lambda_3$ guarantees that this expression always yields a real value. There are two possible values for θ, these are due to a fundamental ambiguity in that a tag tilted away from the viewer will yield the same projected ellipse as a tag tilted, by the same amount, towards the viewer (Figure 3.3). The additional information required to solve this ambiguity must come from elsewhere in the image. Rotation matrices $\mathbf{R}_1$ and $\mathbf{R}_2$ now almost suffice to define the *pose* of the circular tag.

The key problem in the original algorithm at this point is due to ambiguity in the eigenvectors that make up $\mathbf{R}_1$. Forsyth's algorithm calls for the eigenvectors to be normalised (unit magnitude) but this leaves the direction of the vector ambiguous. This means that the algorithm as it stands will unpredictably introduce reflections in the transformation matrices. One manifestation of this is that the calculated normal vector of the tag will point in the opposite direction to the one expected. The sign of each eigenvector varies dependent upon numerical quirks in the routines used to find the eigenvectors of the original matrix. Applying the two transformations

to the unit normal vector $(0\ 0\ 1)^\mathsf{T}$ demonstrates that the z-axis component of the third eigenvector must have a value less than 0 in order for the normal vector to point towards the camera. The direction of the eigenvector should be set to ensure this. It is also important to ensure that the matrix $\mathbf{R}_1$ does not include any reflection of the x- or y-axes. This can be achieved by checking the determinant of the whole matrix. A determinant of -1 indicates that the matrix incorporates a reflection. This can be compensated for by changing the direction of one of the eigenvectors (corresponding to the x or y eigenvalue).

In summary, the steps for normalising $\mathbf{R}_1$ are as follows:

1. If the z-axis component of the third eigenvector ($\mathbf{e}_3$) is greater than zero then multiply the entire eigenvector by -1;
2. If $|\mathbf{R}_1| = -1$ then multiply the first eigenvector by -1.

The pose of the tag does not provide enough information to successfully read data from it. Geometrically, this process finds a rotation of the camera axis which results in the imaged ellipse appearing as a circle. The complete transformation must transform the imaged ellipse into the *unit* circle. To achieve this a scaling and a translation along the x-axis must be incorporated, the parameters of which can be found by multiplying out Equation 3.2 and substituting for θ (notice that the x and y terms in this equation are the same due to the choice of θ above):

$$\mathbf{c}_{RR}^\mathsf{T}\mathbf{R}_2\mathbf{C}_R\mathbf{R}_2^\mathsf{T}\mathbf{c}_{RR} = 0 \tag{3.2}$$

$$\begin{bmatrix} x_{c_{RR}} & y_{c_{RR}} & z_{c_{RR}} \end{bmatrix} \begin{bmatrix} \lambda_2 & 0 & -\alpha \\ 0 & \lambda_2 & 0 \\ -\alpha & 0 & \beta \end{bmatrix} \begin{bmatrix} x_{c_{RR}} \\ y_{c_{RR}} \\ z_{c_{RR}} \end{bmatrix} = 0 \tag{3.3}$$

$$\pm\sqrt{(\lambda_2 - \lambda_1)(\lambda_3 - \lambda_2)} = \alpha \tag{3.4}$$

$$\frac{\lambda_1(\lambda_2 - \lambda_1) + \lambda_3(\lambda_3 - \lambda_2)}{\lambda_3 - \lambda_1} = \beta \tag{3.5}$$

Simplifying and completing the square in Equation 3.3 yields the translation t_x and the scale factor s:

$$\begin{aligned} \left(x_{c_{RR}} - \frac{\alpha}{\lambda_2}\right)^2 + y_{c_{RR}}^2 &= \frac{1}{\lambda_2}\left(\frac{\alpha^2}{\lambda_2} - \beta\right) \\ (x_{c_{RR}} - t_x)^2 + y_{c_{RR}}^2 &= s^2 \end{aligned}$$

Repeating this derivation for a value of $-\theta$ shows that if this value is selected for the pose of the circle, the translation along the x-axis must be in the opposite direction also. Intuitively this is because the rotation has moved the centre of the circle in the opposite direction to before. This gives the final transformation in four-dimensional homogeneous coordinates:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ h_c \end{bmatrix} = \left[\begin{array}{c|c} \mathbf{R}_1\mathbf{R}_2 & \mathbf{0} \\ \hline \mathbf{0} & 1 \end{array}\right] \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \pm t_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ z_t \\ 1 \end{bmatrix}$$

The image coordinates are thus $\mathbf{i} = (x_i, y_i) = (x_c/z_c, y_c/z_c)$.

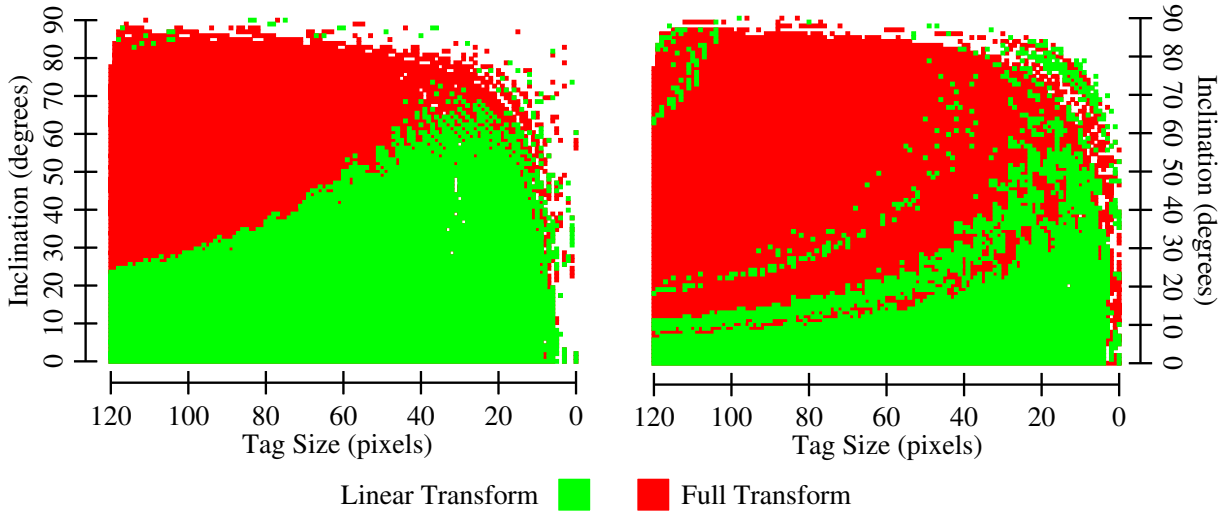


Figure 3.23: Comparing TransformEllipseLinear to TransformEllipseFull for CircleInner (left) and CircleOuter (right) tags

3.6.2 Evaluation

The three-dimensional tag transformation shows significant improvement when attempting to read a tag over existing techniques using the linear perspective approximation. Figure 3.23 shows the maximum angle of inclination for which the tag could successfully be decoded for decreasing tag size in the image. The TransformEllipseLinear algorithm (which closely approximates the approach used in the TRIP location system) performs poorly compared to the TransformEllipseFull algorithm. When a tag is close to the camera the TransformEllipseLinear algorithm fails at large inclinations due to the high perspective effect (as predicted in Figure 3.14). Figure 3.24 compares the performance between square and circular tags. The superior performance of the square tag arises from the difference in size and shape between the datacells on the square and circular tag designs. The limitations on tracking which arise from datacell geometry are investigated in more detail in the next chapter.

3.6.3 Resolving pose ambiguity

The ellipse ambiguity problem means that the TransformEllipseFull algorithm produces two possible valid transformations for the imaged ellipse. There are a number of possible methods for selecting which of the two transforms is correct using additional information in the image.

Location test

The TransformSelectEllipse algorithm chooses between the two possible transformations by estimating the location in camera coordinates of the centre of the tag for both options (the outer edge estimates). These estimates are subsequently compared to the possible locations of the transformations for the inner edge of the tag bullseye (the inner edge estimates). The transform with corresponding outer edge estimate which minimises the Euclidean distance to one of the inner edge estimates is selected as the preferred candidate.

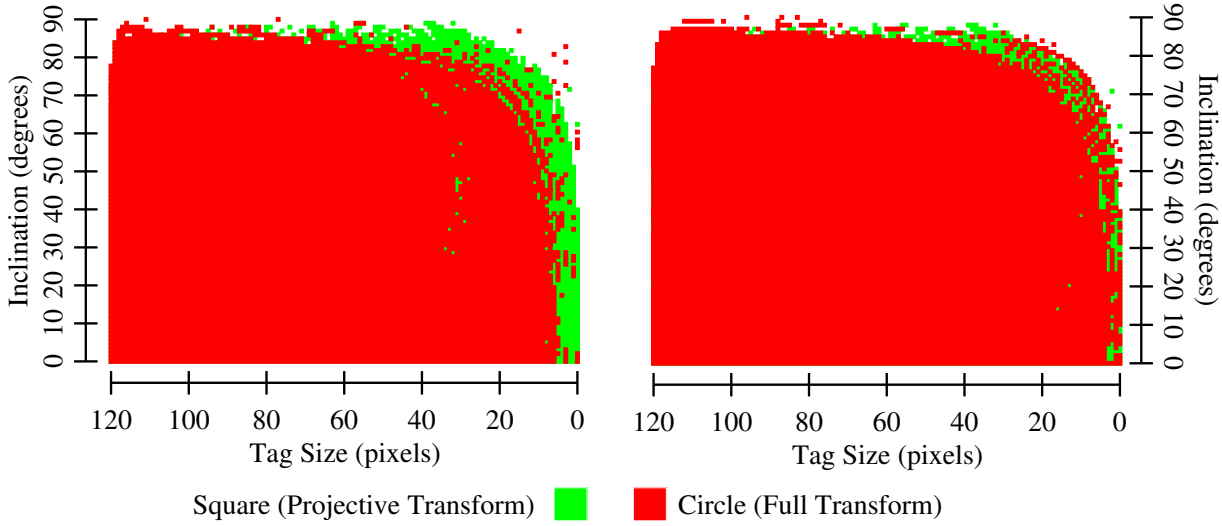


Figure 3.24: Tag decoding performance of square and circular tags: Square and CircleInner tags (left), and Square and CircleOuter tags (right)

Error-of-Fit test

The TransfomSelectEllipseErrorOfFit algorithm utilises both transformation possibilities to estimate a number of points which should lie on the inner bullseye ellipse. An error-of-fit value is computed from the proximity of each set of predicted points to the inner bullseye ellipse. The transformation which produces the smallest error-of-fit value is selected as the preferred candidate. Various functions have been proposed for evaluating the quality of fit of a point to a target ellipse [120]. These provide various approximations to the Euclidean distance (which is expensive to compute). Cantag allows the system designer to select the error-of-fit function and a choose an aggregation method (mean, mode, or maximum value) collecting the results from the hypothesised points into a single score for the candidate transform.

Known point prediction

Another possible approach is to generalise the technique used in the TRIP location system [36]. Known point prediction in the TRIP system utilises a grey-scale corner finding algorithm to find the image (pixel) coordinates of a particular corner of the synchronization sector. Each candidate transform is used to produce an estimate of this point and the candidate producing the most proximate estimate is selected.

The first problem with this approach is that there is no guarantee that the particular corner of interest will actually exist. The particular point of interest is specified to be the outer-corner, moving in a clockwise direction. The sector adjacent to this contains part of the data payload and so can have an arbitrary value (the TRIP coding scheme is systematic). However, if the value of the sector happens to be 2 (the TRIP coding scheme uses ternary symbols), then there will not be a visible corner because the datacell will obscure it.

Secondly, there is no guarantee that the two possible estimates for the candidate point will differ by a significant amount. The pose ambiguity of an ellipse produces two candidate transforma-

tions which are self-similar up to a rotation. If the synchronization sector (and hence the corner point) happens to lie on this axis of rotation then it will display very little displacement.

3.7 Summary

This chapter has introduced Cantag, a computer vision system for tracking fiducial marker tags. Cantag is designed for transparent operation by exposing the intermediate values which contribute to a chosen result. This provides important benefits for dependability by allowing easy investigation into erroneous results. The ComposedEntity class demonstrates a particular implementation strategy for transparent operation using template meta-programming in C++.

The integrated OpenGL test harness in Cantag has been used to achieve a reliable software implementation. Its use has also highlighted algorithmic issues with some of existing MBV approaches. These issues highlight the difficulty in providing a reliable system both in terms of producing an error-free implementation and in acquiring an error-free specification.

Current symbolic coding techniques for data on fiducial marker tags are weakened by the rotational symmetry present in many designs. Rotational invariance provides a useful abstraction for designers by permitting the robust application of existing coding systems to marker tags.

A full three-dimensional back-projection algorithm for circular tags has been described. Its superiority to current techniques for decoding tag data was demonstrated using simulation. A comparison of this algorithm to the performance of square tag designs shows that square tags still demonstrate superior reading performance. This result is investigated in more detail in the next chapter in the course of considering the fundamental limits to the performance of a location system.

Chapter 4

Algorithmic Dependability

Sensor data are inherently an approximation of the environmental input and thus, to some degree or another, all location systems have limitations to their performance and accuracy.

This chapter describes the use of Cantag for investigating the performance of vision-based location systems. A number of measures for assessing the performance of tag designs and tracking algorithms are introduced. Firstly, an information theoretic view of the image's content is refined to define *sample strength*, a metric expressing the readability of a tag. An algorithm for estimating this metric from the recorded image is presented and justified. The location accuracy of the tracking process is examined for various processing options both in simulation and with real-world data. Dependable processing pipelines are identified based on their agreement with the theoretical and simulated models of the system.

4.1 Specifying Performance

Concepts such as the *minimum performance level* [72] specify an upper bound on the location error of an entire deployed system as a scalar value. Systems producing sightings with non-Gaussian distributed error cannot be acceptably summarised with this technique. For example, in Cantag, the pose ambiguity problem results in a bimodal distribution of error for the recovered pose of a tag. Figure 4.1 shows the results of a simulation in Cantag using the OpenGL harness. The error in the normal vector of the tag's pose is shown against increasing tag inclination for tags held in the centre of the image at various distances from the camera. Not only is the error bimodal and hence poorly summarised by the mean of a Gaussian distribution but it is also dependent upon the physical orientation of the tag. Exclusion of either of these factors results in an error value which is grossly pessimistic for the majority of the system coverage.

Other researchers have suggested the use of probability distributions to show the uncertainty of location sightings from a system [7]. This representation is capable of expressing arbitrarily complex error distributions. However, these results must often still be parameterised over the current geometry of the tracked objects. The task of deriving and justifying these probability distributions is non-trivial for complex systems.

The alternative to this approach taken here is to derive metrics which describe specific aspects of system performance derived from the tracking process itself. Defining metrics with reference

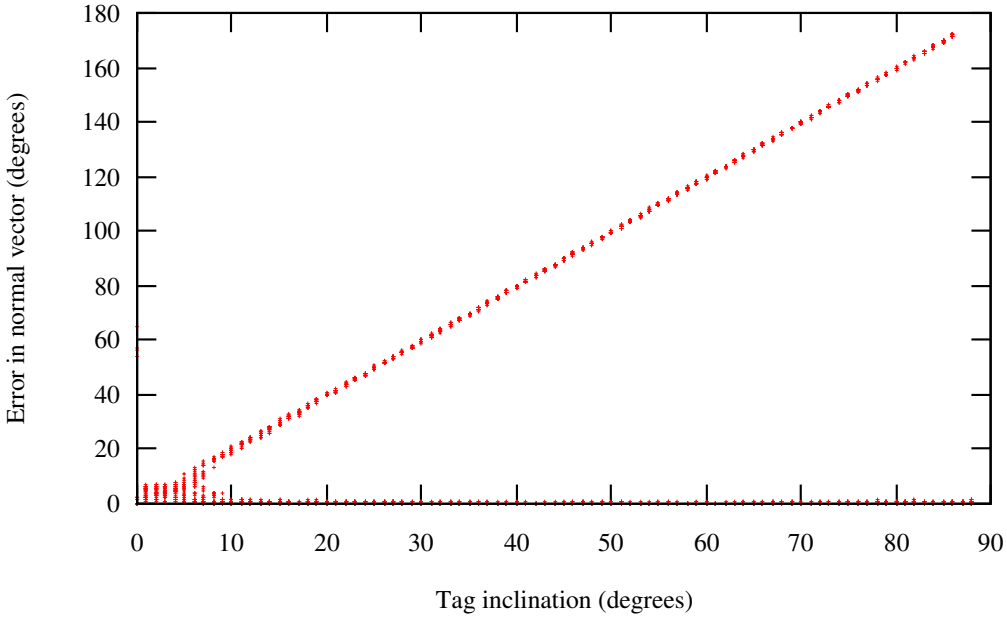


Figure 4.1: The error in the tag's normal vector is bimodal

to the raw sensor data and the particular processing algorithms aids in the identification of significant contributing factors to the metric. For example, the affect of physical tag orientation on the error in the pose estimate (shown in Figure 4.1) is predictable due to the ambiguity evident in the back-projection algorithms (Section 3.6).

4.2 Features of the Camera Model

There are numerous variables and factors describing the configuration of a Cantag pipeline and affecting its operation. However, the effect of altering one of these values is often correlated with changes to some of those remaining. This section details the model used for the operation of the camera and describes the quantity of *full tag size* which unifies many of the correlated factors.

A pin-hole model of operation is adopted for the camera and imaging system. In reality, effects such as lens distortion have a significant effect on the final image but it is assumed that these have been be corrected using the camera correction algorithms discussed in the previous chapter. All distance measurements are given in units of tag widths; if the tag is $10\mathrm{cm}$ wide then there are 10 units in a metre.

The distance between the tag and the camera is inversely proportional to the size of the tag in the image. Figure 4.2 shows a top-down view of a tag parallel to the camera. The width of the projection of the tag is τ/z and the total width of the camera's field-of-view is $2\tan(\theta/2)$ spread over R pixels. The number of pixels occupied by the projection of the tag is $\frac{\tau R}{2z\tan(\theta/2)}$. Thus, the distance between the tag and the camera is inversely proportional to the size of the tag in the image. The constant of proportionality incorporates the field-of-view of the camera (θ), the physical size of the tag (τ units of distance) and the pixel resolution of the camera (R).

Tilting the tag with respect to the camera will change its size in the image. In order to retain the

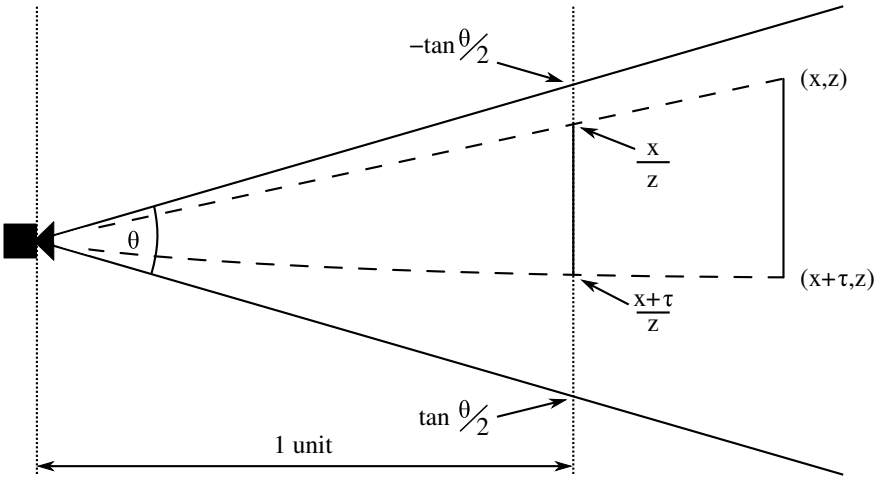


Figure 4.2: Tag size is inversely proportional to distance from the camera

Resolution (pixels)	Field-of-view (degrees)	Full Tag Size (pixels)	Distance (unit)
2614×1958	52.0	50	53.6
2614×1958	52.0	100	26.8
752×480	62.9	50	12.3
752×480	62.9	100	6.2
352×288	44.8	50	8.5
352×288	44.8	100	4.3

Figure 4.3: Full tag sizes for a number of example camera configurations. Distance is given in units of tag width

proportional relationship with the distance from the camera the quantity of *full tag size* is more precisely defined as the number of pixels that the imaged tag would occupy in the image if it were to be parallel to the camera in its current position. Results quoted with respect to *full tag size* may be understood without reference to a particular camera or lens configuration. Figure 4.3 shows some example *full tag sizes* for a selection of cameras.

A comparison between square and circular tags based on the full tag size is effectively comparing tags with the same *bounding boxes* rather than tags which occupy the same *area*. This gives an advantage to square tags which fully utilise the bounding box of the tag. If comparison based on area rather than on bounding box is required then a scale factor must be applied to the full tag size. Equating the occupied area of a square tag with full tag size s and a circular tag with size c yields:

$$\begin{aligned} s^2 &= \pi(c/2)^2 \\ \frac{2}{\sqrt{\pi}}s &= c \\ \frac{2}{\sqrt{\pi}} &\approx 1.1 \end{aligned}$$

This means that, if an area-based comparison is required, a square tag with size s should be compared to a circular tag with size $1.1s$. The comparisons that follow utilise only the full tag

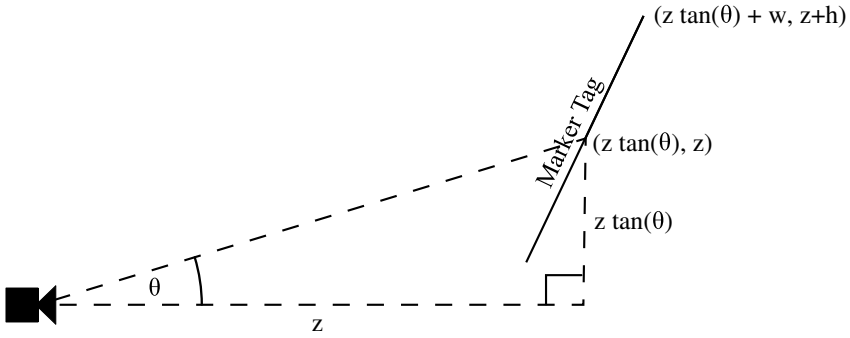


Figure 4.4: Tag size with fixed orientation for positions along a ray. The tag size is linearly related to the inverse of the distance from the camera

size without applying an area-based correction factor. The decision is motivated by a number of observations. Firstly, it is often the case that the size of the tag which is deployed in the environment is limited by the object it is attached to. This object must be larger than the bounding box of the tag in order to guarantee that the tag can be attached. Secondly, it is often the case that the separation between the circular and square designs is more significant than the required scale factor. In the following comparisons between square and circular tags this correction factor would have the effect of moving the results for the two designs closer together by 10%.

Figure 4.4 shows a tag with an arbitrary inclination positioned relative to a ray emanating from the camera. The centre position of this tag is at $(z\tan\theta, z)$ and the corner position is $(z\tan\theta + w, z + h)$, w, and h are constants which encode the orientation and size of the tag. The size of the projected image of this tag is therefore:

$$\left(\frac{z\tan\theta + w}{z + h} - \frac{z\tan\theta - w}{z - h}\right) = 2\frac{w - h\tan\theta}{z + h^2/z} \tag{4.1}$$

This result shows that, for a particular fixed inclination (as specified by w and h) when the size of the tag is small compared to the distance from the camera ($h^2/z \approx 0$), the inverse of the tag size varies linearly with the distance from the camera as the tag is moved along any particular ray. This fact is used in the next section to justify the use of linear interpolation on quantities linearly related to the tag size.

4.3 Sample Distance

Successfully decoding the data stored on a tag becomes more difficult as the size of the tag decreases in the image. A fundamental upper limit on read performance of the system may be established by considering the *sample distance* for each datacell on the tag. The sample distance for a datacell refers to the minimum distance from the projection of the cell's centre to the cell edge. Figure 4.5 shows the sample distance for all the datacells of three example tags. The radius of the circle drawn in each cell shows the sample distance for that cell. Comparison of the first tag (with 2 rings and 18 sectors) and the second tag (with 2 rings and 24 sectors) shows the sample distance vector (where the circle intercepts the datacell edge) changing from a radial direction to a tangential direction as more sectors are added.

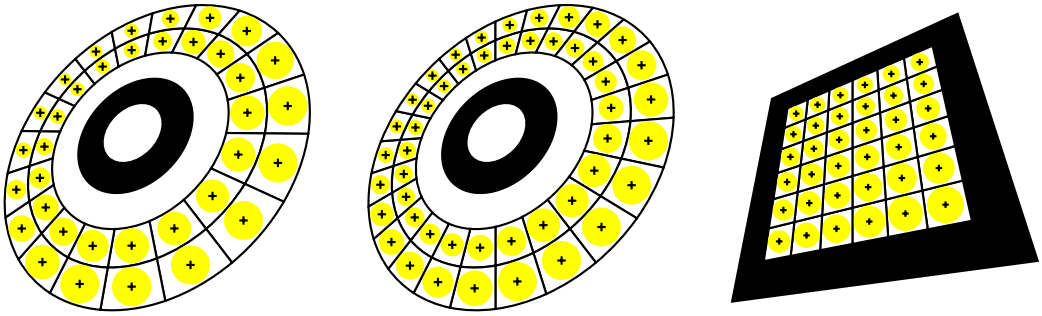


Figure 4.5: Sample distances for three example tags. The radius of each circle shows the sample distances for the containing datacell

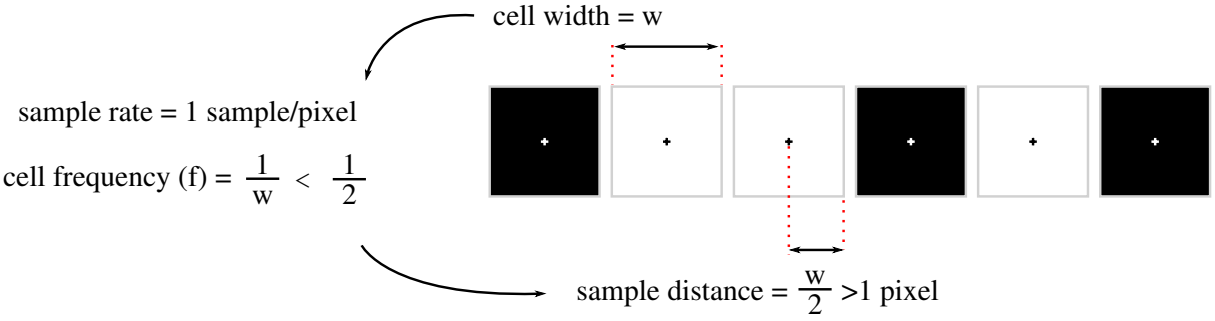


Figure 4.6: Analogy to the Nyquist-Shannon sampling limit

If the sample distance for a datacell is less than one pixel then there is the possibility for the system to read the value of an adjacent cell rather than the target cell and for a bit-error to occur. This situation is analogous to the Nyquist-Shannon sampling theorem which states that a discrete representation of an analogue signal is only possible if the highest frequency component of the analogue signal is less than half the sampling rate. Consider a stream of datacells each with a width of w pixels. The frequency of these cells is therefore $1/w$ cells per pixel. The sampling rate is 1 sample per pixel and so the sampling theorem requires that $1/w < 1/2$ or $w > 2$. This corresponds to sample distance of 1 pixel (Figure 4.6).

If the sample distance is not greater than 1 pixel then a bit-error can be introduced into the tag payload. If an error-correcting code has not been used on the tag then the tag will fail to be read successfully after the first bit-error i.e. a failure is expected when the smallest sample distance for any datacell on the tag falls to 1 pixel. The smallest sample distance for any datacell on the tag is referred to as the *minimum sample distance*.

Figure 4.7 shows how a tag with minimum sample distance of 1 pixel (cell width of 2 pixels) will alias onto a pixel array at 45 degrees. The positions of the ideal sample points (shown as filled circles) show that there is still sufficient information in the image to recover the data.

The minimum sample distance will vary linearly with the size of the tag in the image. The distance such that the minimum sample distance is 1 pixel can be computed by exploiting the fact that tag size is linearly related to distance from camera (Equation 4.1). A number of minimum sample distance values (at significant distance from the camera) may thus be combined by linear interpolation to discover the distance from the camera where the minimum sample distance is 1 pixel—this is referred to as the *interpolated sample distance*. The full tag size when the tag

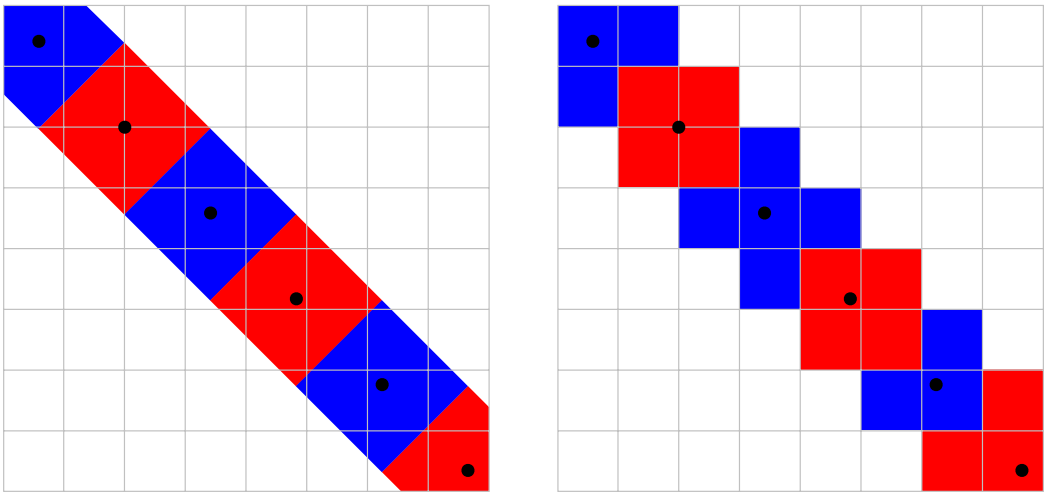
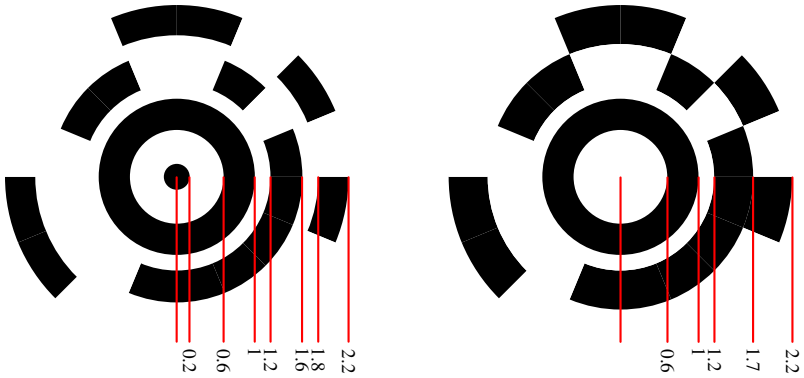


Figure 4.7: Pixel aliasing affect on minimum sample distance



(a) Original TRIP Tag (b) Cantag equivalent

Figure 4.8: Approximating the original TRIP tag design

is positioned at the interpolated sample distance is referred to as the *interpolated tag size*.

4.3.1 Circular tag design

The sample distance measure provides a means for quantifying the data-carrying aspect of the tag design. Tag designs with a large sample distance measure are easier to read than designs with a smaller distance. Figure 4.8 shows the Cantag equivalent of the original TRIP tag design [35]. Cantag does not require a gap between each data ring and so this design has slightly thicker data rings than the original TRIP tag. Figure 4.9 shows the interpolated tag size for the TRIP tag design. The interpolated sample distance resulting from the best performing combination of rings and sectors was recorded for increasing payload size of interest. For example, for small payload sizes two ring tags provide the best performance. However, if the desired payload is not a multiple of two then it is not possible to use a two ring tag and so other, less efficient, options must be used—this is the reason why it is occasionally suitable to select a three ring tag at the low end of the graph.

The flat portion of each curve corresponds to payload sizes for which adding additional data

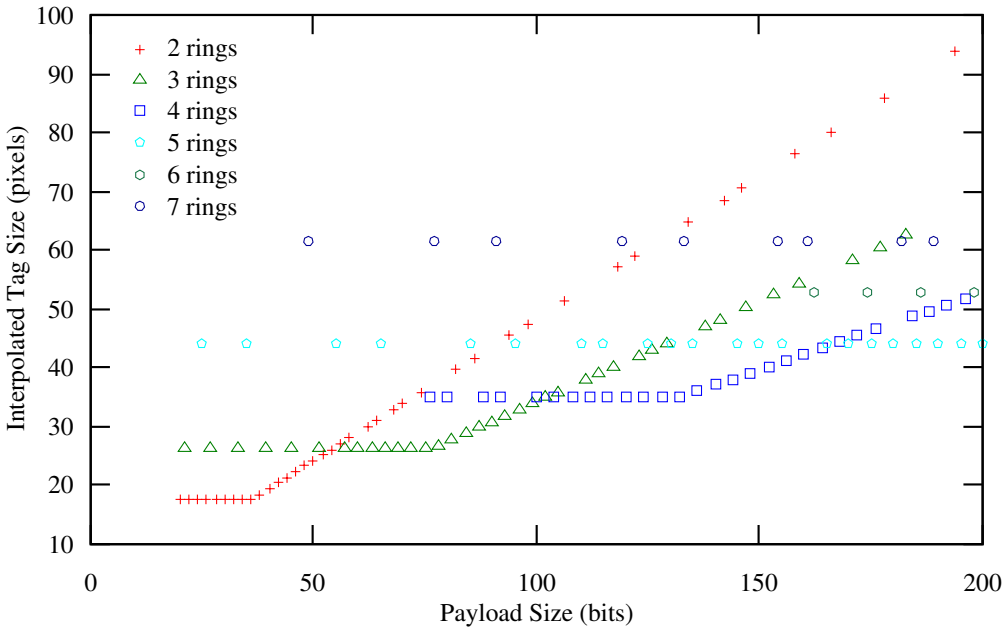
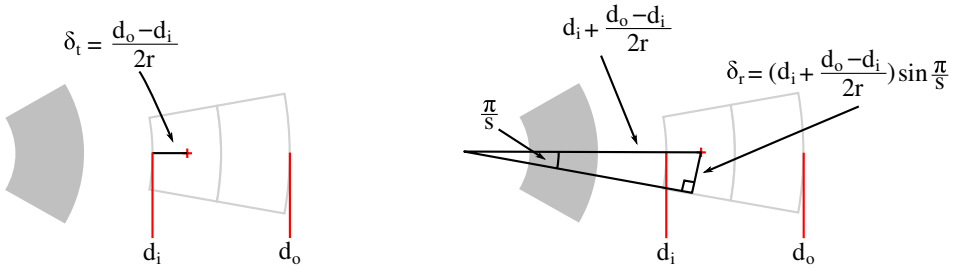


Figure 4.9: TRIP tag interpolated tag size for increasing payload size

causes no change in the interpolated tag size. For example, according to this model the performance of a 68-bit, four ring tag is expected to be the same as the performance of a 120-bit, four ring tag. As the minimum sample distance is a worst-case metric it is possible that the 68-bit tag would perform better in reality for some proportion of the time. For these tags the radial distance between rings is much less than the tangential distance between sectors. Therefore the addition of another sector to the tag does not reduce the sample distance because the radial distance remains the limiting factor. The original TRIP design with 2 rings and 8 sectors could have been extended to contain 2 rings and 17 sectors without sacrificing the ability to decode the tag.

These observations may be utilised to provide a systematic means for selecting the proportions of the tag to allocate to the data bullseye. The ideal layout would produce datacells with equal distance from the centre of the cell to any edge. For a circular tag, the best approximation to this is to equalise the radial and tangential distances for the cells. Unfortunately, this is not possible either because the cells on the outer rings have greater radial size than cells on the inner rings. The best choice is to equalise the radial and tangential lengths for the cells on the inner data ring because this will ensure that the radial width of each data ring is as small as possible. This, in turn, minimises the radial width of the sectors on the inner ring. If instead, one were to equalise the two sizes of the outer data ring this would result in a larger width for each data ring, which would in turn move the inner edge of the data ring closer to the centre of the tag, which would result in reducing the radial width of the cells on the inner data ring making them harder to read. Since the inner data ring cells are the limiting factor on the tag this is obviously a bad choice.

The minimum sample distance measure which forms the basis of the model used for this optimisation argument does not take into account the size of the target bullseye when tracking the tag. Successful designs must seek to maximise the size of this feature in order to improve the chances of the system even noticing the tag in the first place. The requirements of the optimal tag design are therefore as follows:



(a) Tangential Distance (b) Radial Distance

Figure 4.10: Tangential and radial size calculation

1. The design should maximise the minimum sample distance of the datacells;
2. The design should maximise the size of the target bullseye;
3. Separation between the edges of the payload area and the target bullseye should be the same size as the datacells.

This final requirement arises from the need to ensure that the targeting features of the tag do not merge into the data area of the tag in the acquired image. It is assumed that if the tag is to be successfully read the system must be able to distinguish features of the size of a single datacell and so distinct features on the tag should be given, at least, this separation.

It is now possible to derive the optimum choice of b_i, b_o, d_i, d_o for a given number of rings (r) and sectors (s) for each type of circular tag design. These values are given as a proportion of the entire tag size—a value of 1 indicates that the feature occupies the entire radius of the tag.

Figure 4.10 shows the calculation of the radial size (δ_r) and the tangential size (δ_t) of a datacell dependent upon the size of the data rings (d_i,d_o) and the number of rings and sectors (r,s).

$$\delta_t = \frac{d_o - d_i}{2r} \tag{4.2}$$

$$\delta_r = \left(d_i + \frac{d_o - d_i}{2r}\right)\sin\frac{\pi}{s} \tag{4.3}$$

The radial and tangential distances are balanced by setting $\delta_t = \delta_r$:

$$\frac{d_o - d_i}{2r} = \left(d_i + \frac{d_o - d_i}{2r}\right)\sin\frac{\pi}{s} \tag{4.4}$$

$$\frac{d_o}{2r} - \frac{d_i}{2r} = d_i\sin\frac{\pi}{s} + \frac{d_o}{2r}\sin\frac{\pi}{s} - \frac{d_i}{2r}\sin\frac{\pi}{s} \tag{4.5}$$

$$\frac{d_o}{2r}\left(1 - \sin\frac{\pi}{s}\right) = d_i\left(\sin\frac{\pi}{s} - \frac{\sin\frac{\pi}{s}}{2r} + \frac{1}{2r}\right) \tag{4.6}$$

$$d_o\frac{1 - \sin\frac{\pi}{s}}{(2r-1)\sin\frac{\pi}{s} + 1} = d_i \tag{4.7}$$

$$\alpha = \frac{1 - \sin\frac{\pi}{s}}{(2r-1)\sin\frac{\pi}{s} + 1} \tag{4.8}$$

The quantity α defines the optimum ratio between the outer and inner edges of the data area. This scale factor provides sufficient information to derive b_i, b_o, d_i, d_o, and w (the width of an individual data ring) for each of the three types of circular tag.

Optimal CircleInner tags

CircleInner tags have the target bullseye entirely inside the data payload area of the tag. The following constraints derive the width of each individual data ring (w) to create a bullseye with maximally large radius for both the inner (b_i) and outer (b_o) edges whilst maintaining the minimum feature size.

$$
\begin{aligned}
d_o &= 1 && (4.9)\\
d_i &= \alpha d_o && (4.10)\\
w &= \frac{1}{r}(d_o - d_i) && (4.11)\\
b_o &= d_i - w && (4.12)\\
b_i &= b_o - w && (4.13)\\
b_i &\geq w/2 && (4.14)
\end{aligned}
$$

If the resulting value for b_i is less than $w/2$ then the innermost edge of the bullseye circle is deemed too small for recognition. In this situation it is not possible to generate an optimal tag layout for the chosen combination of rings and sectors.

Optimal CircleSplit tags

CircleSplit tags have the data payload area overlaid on top of the target bullseye. This layout maximises the radius of the outer bullseye edge and places the outer edge of the data ring as close as possible to this.

$$
\begin{aligned}
b_o &= 1 && (4.15)\\
d_o &= 1 - w && (4.16)\\
d_i &= \alpha d_o && (4.17)\\
w &= \frac{1}{r}(d_o - d_i) && (4.18)\\
&= \frac{1}{r}(1 - w)(1 - \alpha) && (4.19)\\
&= \left(\frac{r}{1 - \alpha} + 1\right)^{-1} && (4.20)\\
b_i &= d_i - w && (4.21)\\
b_i &\geq w/2 && (4.22)
\end{aligned}
$$

Again, if b_i is less than $w/2$ then there is no optimal layout available for the chosen combination of rings and sectors.

Optimal CircleOuter tags

CircleOuter tags have the target bullseye completely outside the data payload area.

$$
\begin{aligned}
b_o &= 1 && (4.23)\\
b_i &= b_o - w && (4.24)\\
d_o &= b_i - w && (4.25)\\
d_i &= \alpha d_o && (4.26)\\
w &= \frac{1}{r}(d_o - d_i) && (4.27)\\
&= \frac{1}{r}(1-\alpha)(1-2w) && (4.28)\\
&= \left(\frac{r}{1-\alpha} + 2\right)^{-1} && (4.29)\\
d_i &\geq w/2 && (4.30)
\end{aligned}
$$

The width of each data ring (w) forms a progression between tag designs. It is not clear, however, how (or if) this progression might be continued for other circular tag designs.

$$\left(\tfrac{r}{1-\alpha} + 0\right)^{-1} \quad \left(\tfrac{r}{1-\alpha} + 1\right)^{-1} \quad \left(\tfrac{r}{1-\alpha} + 2\right)^{-1}$$

CircleInner CircleSplit CircleOuter

The minimum sample distance a the CircleInner tag is shown in Figure 4.11. Significant improvement is shown over the performance of the original TRIP layout (Figure 4.9). The flat regions of the performance curves which were evidenced by the TRIP tag design are no longer present. A tag at the beginning of one of the flat regions has the same performance as tags with more data bits and the end of the region. This means that the tag at the beginning of the flat region was not making the most efficient use of the available data space—fewer datacells should allow more space for each cell and thus increase the minimum sample distance.

As expected the interpolated tag sizes of the CircleSplit and CircleOuter tags are inferior to the sizes for CircleInner tags. Figure 4.12 compares the performance of the CircleInner tag with the CircleOuter, CircleSplit and TRIP tags. Negative values indicate that the tag of interest had a larger interpolated tag size (and thus poorer performance) than the CircleInner tag. The CircleSplit tag requires a tag approximately four pixels larger than the CircleInner tag and the CircleOuter tag requires a tag eight pixels larger than the CircleInner tag. This is because, at the smallest size the radius of each ring of the tag will be 2 pixels (corresponding to a sample distance of 1 pixel). Thus, for a given minimum size of a CircleInner tag, the equivalent CircleSplit tag must have an additional 2 pixels all the way round the edge for the outer black border. This increases the width by 4 pixels. The CircleOuter tag has a white ring followed by a black ring around the data area and so incurs an additional 4 pixel cost.

Figure 4.12 also compares the CircleInner tag to the performance of the original TRIP tag design. The designs periodically converge at the points where the CircleInner design selects

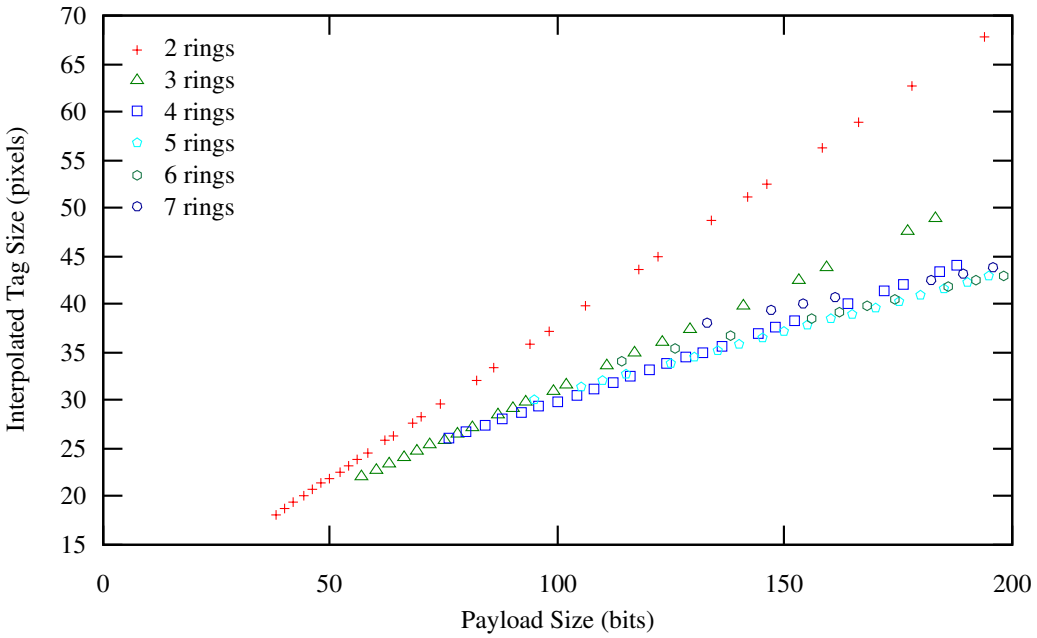


Figure 4.11: Interpolated tag size of CircleInner tags against payload size

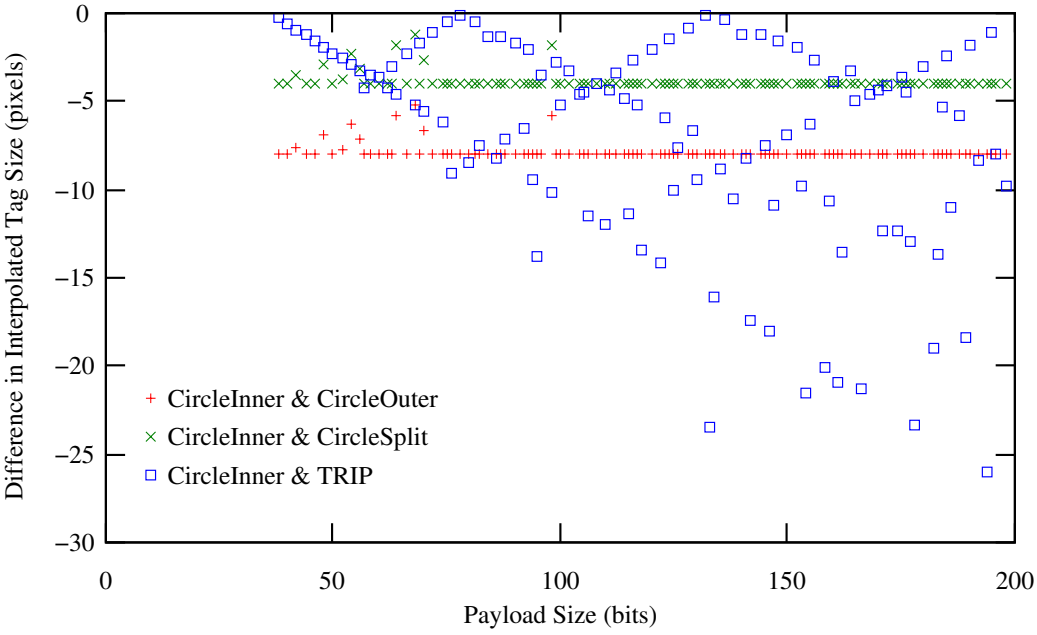


Figure 4.12: Comparing CircleInner tags with the remaining circular designs

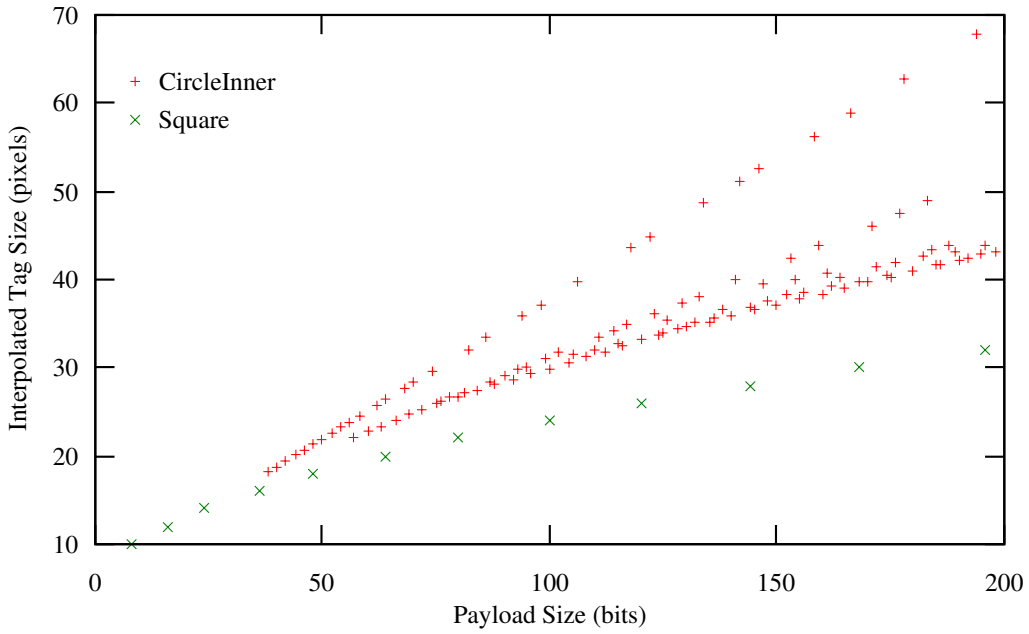


Figure 4.13: Comparing CircleInner and Square tags

the same value for the bullseye inner radius as used by the TRIP tag. However, it can be seen that the variation in performance grows (at an approximately linear rate) as the payload size increases.

The minimum sample distance indicates that the highest performing circular tag designs optimally balance the radial and tangential sizes of the datacells. Analysis of the design used in the TRIP system suggests that the tag could have incorporated many more datacells without any drop in worst-case performance. The optimised layout scheme consistently performs better than the static fixed tag layout used in the TRIP system.

4.3.2 Comparing square and circular tags

The minimum sample distance measure can also be used to gain insight into the relative performance of square and circular designs. Figure 4.13 compares interpolated tag size between the Square and CircleInner tags with increasing payload sizes. For a square tag, the addition of another datacell entails moving from $n \times n$ to $(n+1) \times (n+1)$ rows and columns. This causes a linear decrease in the size of each cell because the tag area is split into $n+3$ portions rather than $n+2$ (recall that the 2 border cells are used for the localisation feature of the tag). However, the linear decrease in cell size results in a quadratic increase in the payload size—this explains the quadratic shape for the Square payload sizes. The Square tag outperforms the CircleInner tag due to its more efficient use of the available tag area for storing the payload. The results compare designs with the same width rather than designs with the same area and so square tag designs have an initial advantage because they fully occupy the allocated space.

Figure 4.14 shows how the minimum sample distance varies for various orientations and positions of a tag. The tag of interest was moved along the ray from the camera origin through the centre of each of nine portions of the image. For each ray the data in the figure is parameterised over θ and ϕ which describe the angles between the normal vector and the camera vector

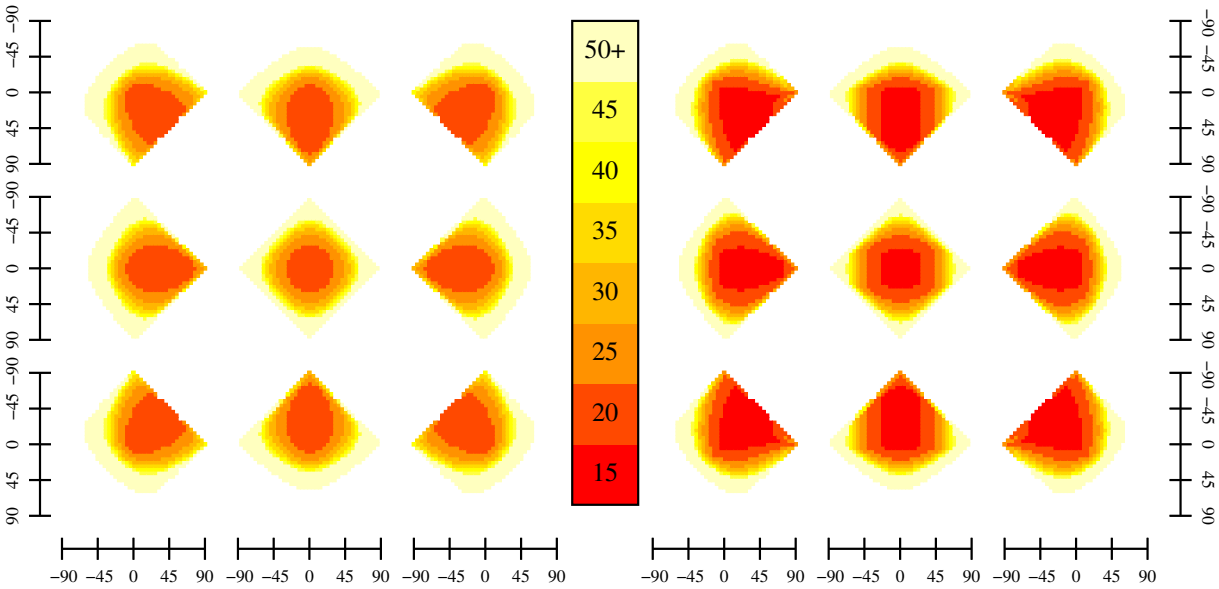


Figure 4.14: The effect of position and pose on interpolated tag size for CircleInner (left) and Square (right)

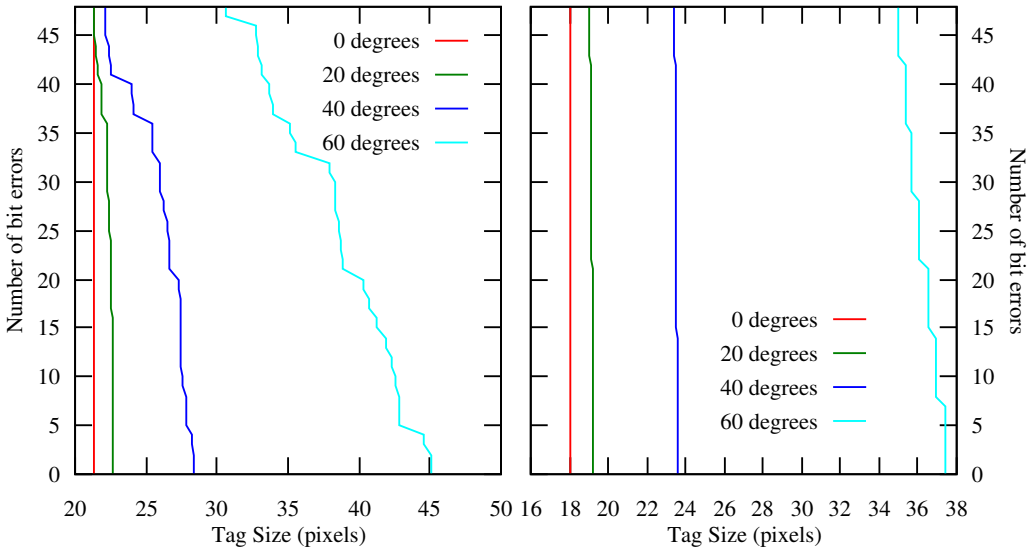
$(0, 0, -1)$ in the x and y directions respectively. The value at each point on the graph is the interpolated tag size. The smaller the interpolated tag size the smaller the tag can be made before it cannot be read and hence the easier the tag is to read. The figure shows data for a CircleInner tag with 48 bits (2 rings with 12 sectors) and a Square tag with 48 bits (rotational invariance means that the centre datacell of the 7 rows and columns cannot be used). It can been seen from the figure that tags positioned off-centre to the camera are easier to read when tilted towards it than when fully facing. As predicted from the curve in Figure 4.13 the Square tag performs better than the Circular tag by generally being readable for smaller interpolated tag sizes.

The effect of the shape of the datacells is evident in the way that the performance of the tags drops off as the tag inclination is increased. The high degree of rotational symmetry possessed by the Circular tag means that when the tag is in the centre of the image the degradation in performance is only dependent upon the angle between the normal vector and the camera vector. The square tag is more directionally sensitive; the square edges of this shape are due to the fact that tilting the tag in the x direction will reduce all the cells in the far edge row in size. Subsequently tilting in the y direction will not reduce the minimum distance of these cells until the tilt exceeds that applied in the x direction.

These results show that square tags are easier to read than circular tags even when the tag is not fully facing the camera. However, the higher level of symmetry in the results for the circular tag are an important feature from a dependability point-of-view. A dependable system should have predictable behaviour and so the fewer variables that influence the tag's performance the better.

4.3.3 Error-correcting coding schemes

The use of rotational invariance has allowed the application of cyclic coding techniques to fiducial marker tags. This permits the introduction of error-correcting codes to the tag design.



(a) CircleInner (b) Square

Figure 4.15: The number of unreadable datacells for selected tag inclinations

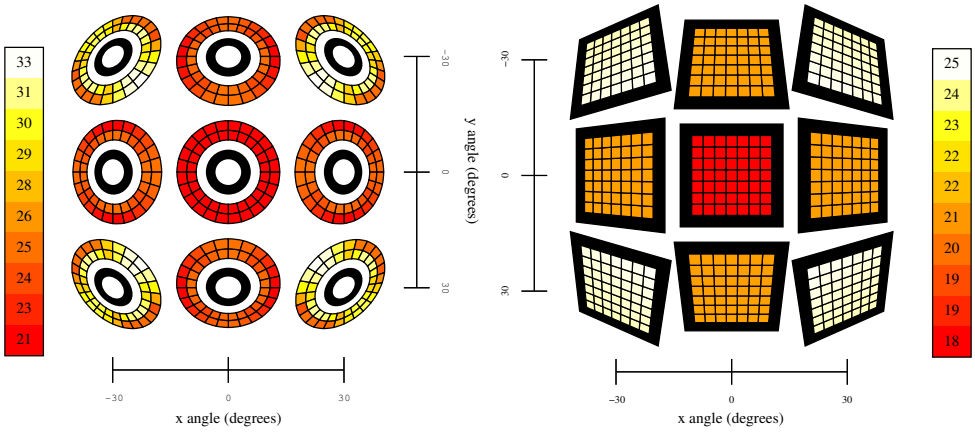
In addition to compensating for image noise it is possible that error-correction techniques might also correct errors from misread datacells caused by the position and pose of the tag. Sample distance allows a quantifiable examination of this hypothesis.

Figure 4.15 shows how the tag size affects the number of datacell errors for a number of possible orientations for 48-bit CircleInner (left) and 48-bit Square (right) tags. Both square and circular full facing (0 degrees) tags transition from no datacell errors to complete failure (all cells invalid) in the space of less than one pixel of tag size. In this situation an error-correcting code will be of little benefit. The rate of decay in the number of datacells successfully read falls as the inclination of the tag increases. For circular tags, a code capable of correcting 15 bits of error would improve the minimum tag size at 60 degrees by approximately 5 pixels. Square tags show a less significant gain—the same error correction only produces a gain of approximately 1 pixel.

Figure 4.16 shows, for each datacell on a tag, the full tag size such that the sample distance is 1 pixel. The figure shows the pixel size for a square and a circular tag held in the centre of the image with various orientations. This diagram indicates the systematic nature of the data errors. Rotational invariance requires that circular tags must be read radially; recall Figure 3.16(b). Therefore, these errors manifest as burst errors in the sampled data. Coding schemes such as Reed-Solomon codes (used for CDs and DVDs) cope well with these sorts of errors. The distribution of errors on square tags is such that the cells on the edge of the tag tilted away from the camera perform least well. As shown in Figure 3.18, rotational invariance permits two possible read orderings for square tags. However the ordering which reads the edges of the tag sequentially is therefore preferable.

4.3.4 Implications for system deployment

The upper performance bound given by the minimum sample distance allows designers to make estimates of the expected coverage of a given deployment of a system. This measure answers



(a) CircleInner (b) Square

Figure 4.16: The required tag size, per datacell, for one pixel of sample distance

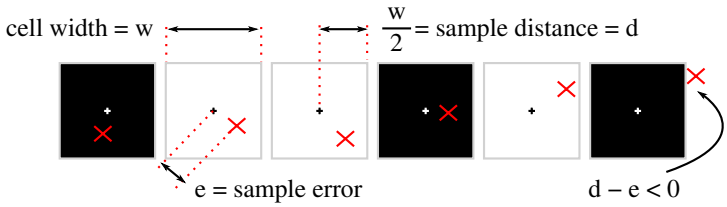


Figure 4.17: Sample error between the ideal and estimated sample points

the question as to whether a particular set of requirements can be met. It also allows a set of tag position and poses to be ranked in terms of ease of reading. Also, the larger the minimum sample distance the more robust the system is expected to be.

However, due to the abstract nature of the model it is expected that effects due to the choice of processing algorithm and image noise will mean that the minimum sample distance is only a bound on performance.

4.4 Sample Error

The sample distance measure gives a theoretical indicator of the margin for error when sampling the data on the tag. It is also important to consider a further quantity: *sample error*. When attempting to read the value of a particular datacell, the sample error is the distance from the ideal sample point to the sample point estimated from the image processing result (Figure 4.17). If the sample error exceeds the sample distance for a particular datacell then it is possible to read an incorrect data value.

Sample error summarizes the performance of the algorithms in the image processing pipeline. A pipeline incorporating robust, noise tolerant, accurate algorithms will produce smaller sample errors than more lightweight processing options. The *maximum sample error* for a tag is used to summarize the worst-case performance.

Figure 4.18 provides a comparison between the sample error derived from the FitEllipseLS and

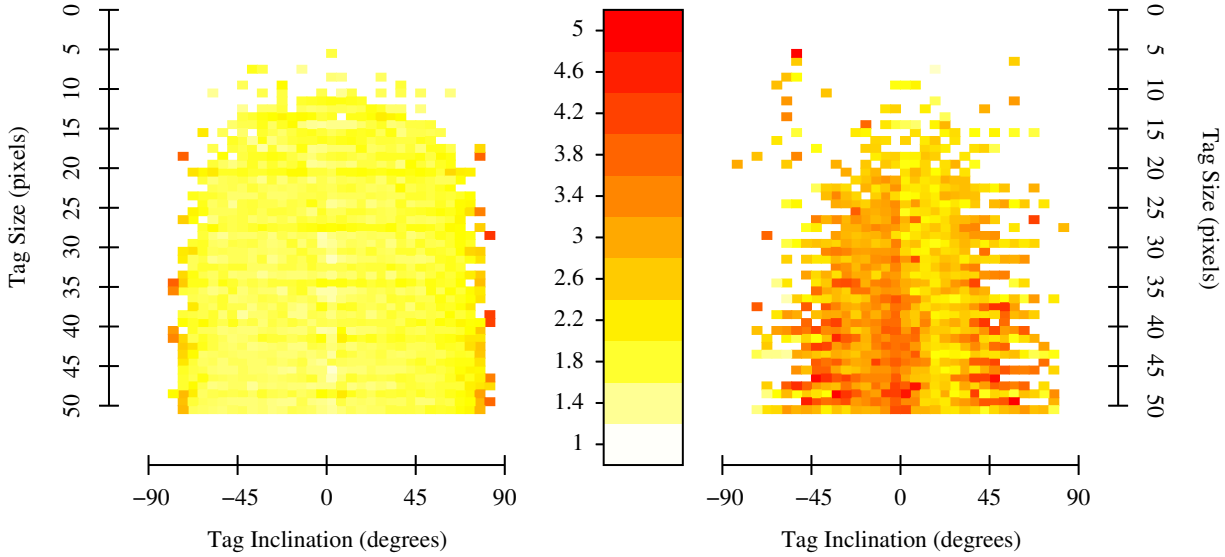


Figure 4.18: Maximum sample error from FitEllipseLS (left) and FitEllipseSimple (right)

FitEllipseSimple algorithms for a CircleInner tag. The image of a tag held in the centre of the image was synthesised (using the OpenGL image source) for increasing distance from the camera and varying rotation about the tag's vertical axis. The figure shows the full tag size on the vertical axis and the tilt of the tag along the horizontal. The FitEllipseLS algorithm shows much less variation in sample error than the FitEllipseSimple algorithm. The datacells on a CircleInner tag lie outside the bullseye contour which is used for recognition. Thus, estimation of the sample point for each datacell requires an extrapolation from the original contour. Decreasing the distance from the contour to the datacell will also decrease the absolute error in the estimated point. This explains why it is possible for the error to remain roughly constant for the FitEllipseLS algorithm despite the increasing distance from the camera: as the tag reduces in size the estimate of the contour from the image will become less accurate due to pixel truncation. However, the reduction in size also decreases the distance between the contour and each datacell. The FitEllipseSimple algorithm is more susceptible to the errors from pixel truncation and so this effect dominates the result. It can also be seen that the FitEllipseSimple algorithm performs more poorly for a tag fully facing the camera (0 degrees tilt) than for a tag with a slight tilt applied. This is due to the nature of the algorithm which searches for a major and minor axis of the hypothesised ellipse—a problem that is ill-posed for a circle.

4.5 Sample Strength

The *sample strength* is defined as the minimum sample distance minus the maximum sample error. If this quantity is positive (i.e. the distance from the sample point of any datacell to the cell's edge is greater than the maximum error in estimating the sample point) then it is expected that the tag will be read successfully. The sample strength also indicates the degree of tolerance that the particular situation has to other errors which are not modelled (such as camera distortion and lighting artefacts). Configurations with a large sample strength are expected to be easier to read than those with a small sample strength. If the sample strength is below zero then there is not sufficient information in the image to decode the tag even in the most favourable conditions.

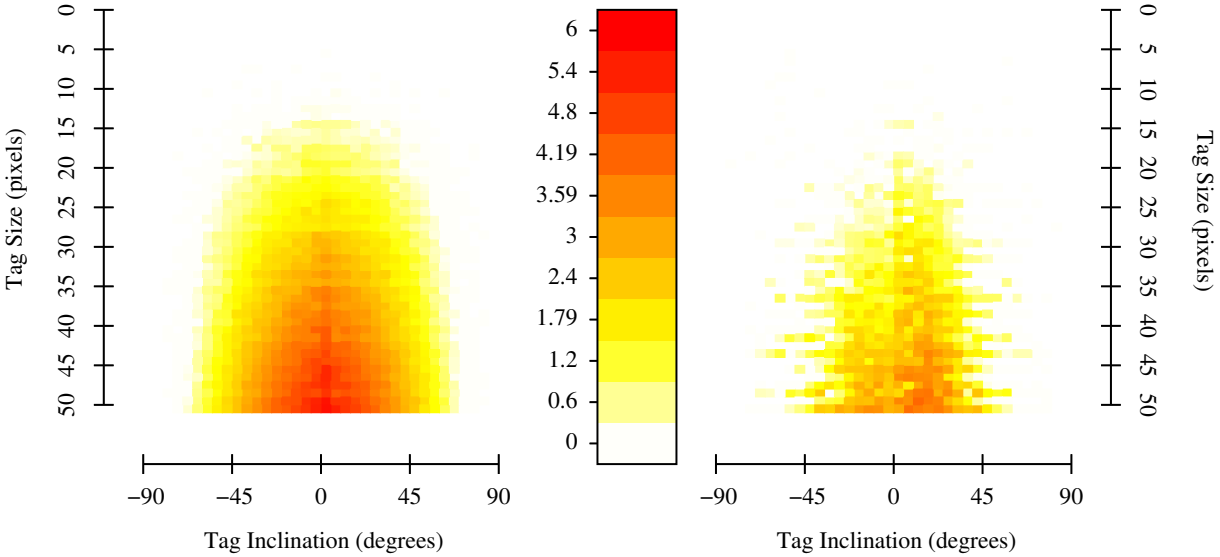


Figure 4.19: Sample strength of FitEllipseLS (left) and FitEllipseSimple (right)

Figure 4.19 shows the sample strength for the FitEllipseLS and FitEllipseSimple algorithms. The region surveyed is the same as in Figure 4.18. Conversely to the previous measure, a large sample strength indicates good performance of the processing pipeline. Despite having relatively constant sample error, the FitEllipseLS algorithm shows degraded sample strength as the tag moves away from the camera—the error remains the same but the target area is reduced in size. The poor performance of the FitEllipseSimple algorithm for high inclination or small tags is exacerbated by the small minimum distance measure for these positions.

4.5.1 Estimated sample strength

An important function of a dependable location system is to be able to estimate the reliability of the produced measurements. The sample strength is a useful value to report in this respect: a small value informs users that the sighting was marginal and might be difficult to reproduce. Similarly, administrators deploying applications may use the measure to evaluate the system coverage and potential improvements to it.

It is not possible to directly measure the sample strength because the quantity depends on knowing the absolute position and pose of the tag in camera coordinates. However, this value can be estimated by measuring the shortest distance from any estimated sample point to the edge of the sampled datacell. The algorithm proceeds by beginning a search from each estimated sample point in the image and finding the closest point with the opposite colour to the colour at the estimated sample point. For example, if the estimated sample point is in the middle of a white datacell then the search must find the closest point in the image with a black value. This method produces an estimate of the sample strength only using quantities directly measured from the input image.

There are a number of sources of error which will affect the quality of this estimate.

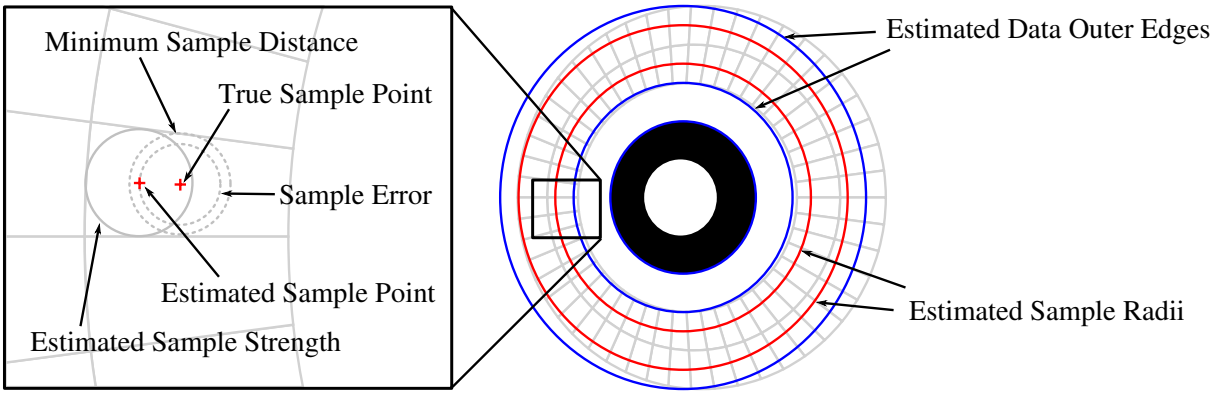


Figure 4.20: Approximation Error when estimating Sample Strength

Approximation error

The shape of the datacells and the effect of perspective distortion will mean that the minimum sample distance is a pessimistic estimate of the size of the datacell. In many cases an error greater than the minimum sample distance will still result in a sample point inside the datacell. Figure 4.20 shows a CircleInner tag tilted around the vertical axis. The TransformEllipseLinear algorithm has been used to estimate the sample points—the estimated sample points lie on the radius of the estimated sample radii marked on the diagram. The systematic error introduced by the TransformEllipseLinear algorithm displaces the estimated sample point radially outwards from the true sample point. The sample strength is the difference between the radii of the circles denoting minimum sample distance and sample error. This is considerably smaller than the estimated sample strength.

Approximation error will result in estimates of the sample strength which are overly optimistic (too large) compared to the true sample strength. Image processing pipelines which commonly do not introduce these pathological errors should be favoured in order to minimise this problem.

Transition error

The estimation algorithm searches every datacell of the tag looking for the minimum distance from the sample point to the edge of the cell. Successful detection of the minimum distance relies on the presence of a transition between the current datacell and any adjacent feature that defines the cell edge. The adjacent feature might be another datacell or might be some other part of the tag template such as the target bullseye or border. If there is no transition the cell edge will not be detectable and so the estimation algorithm will select a minimum distance which is larger than the true value.

Transition error will also result in estimates of sample strength which are overly optimistic (too large). Coding schemes and tag designs which produce many transitions on the tag payload will help to minimize this error. The CircleOuter tag design is promising from this respect because the data ring is surrounded by a ring of white followed by a ring of black. This places a bound on the maximum distance that any search can proceed before experiencing a transition.

Sample cell error

If the estimated sample point lies outside the target datacell then the sample strength for the tag is negative and it is no longer guaranteed that a correct reading of the data will occur. In this situation the search algorithm is begun in the wrong datacell but will still produce a positive value for the estimated sample strength.

Sample cell errors will produce optimistic estimates of sample strength. Error-detecting coding schemes reduce the probability of the system failing to notice an invalid reading and so only utilising estimated sample strengths for valid tags will alleviate this problem.

Verification

Verification of the estimation algorithm is performed by running a simulation using the OpenGL test harness in Cantag. The ground-truth values used as input for the simulation are used to calculate the minimum sample distance and maximum sample error, thus yielding the actual sample strength. This value can be compared with the result from the estimation algorithm.

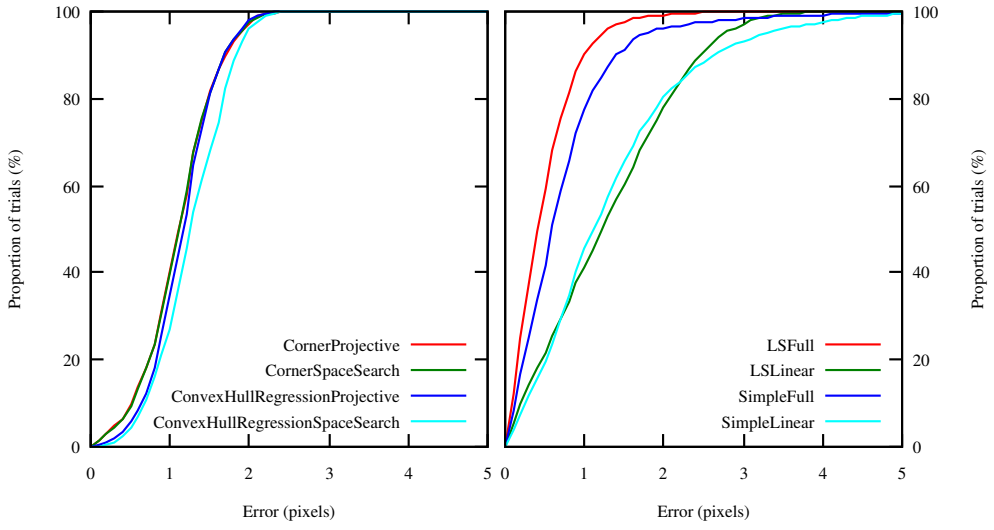
Figure 4.21 shows the cumulative error curves for the three types of circular tag and a square tag design. Data were collected for tags with a uniform range of angles, image position and distance from camera. The graph shows the percentage of locations tested which exhibited fewer than a given number of pixels of error between estimated and actual sample points. Only those locations such that all processing pipelines successfully decoded the tag are included in the statistics.

Figure 4.21(a) shows error results for the Square tag design with a variety of processing pipelines. Almost all locations produce an error below 2 pixels regardless of the processing algorithms chosen. It can be seen from comparison with the remaining graphs that a circular tags using the TransformEllipseFull algorithm produce lower error estimates than the Square tag design. CircleOuter and CircleSplit tags produce largely similar results which are superior to the CircleInner tag. This is because the CircleOuter and CircleSplit designs both ensure a colour transition in the radial direction. The CircleOuter tag places a white ring followed by a black ring outside the data rings, whereas the CircleSplit tag places a black ring followed by a white background outside the data rings. This guarenteed transition places a bound on the transition error which can be produced by the estimation algorithm. The CircleInner tag has only a white background outside the data rings and so does not constrain the search size when the algorithm is searching for a white to black transition.

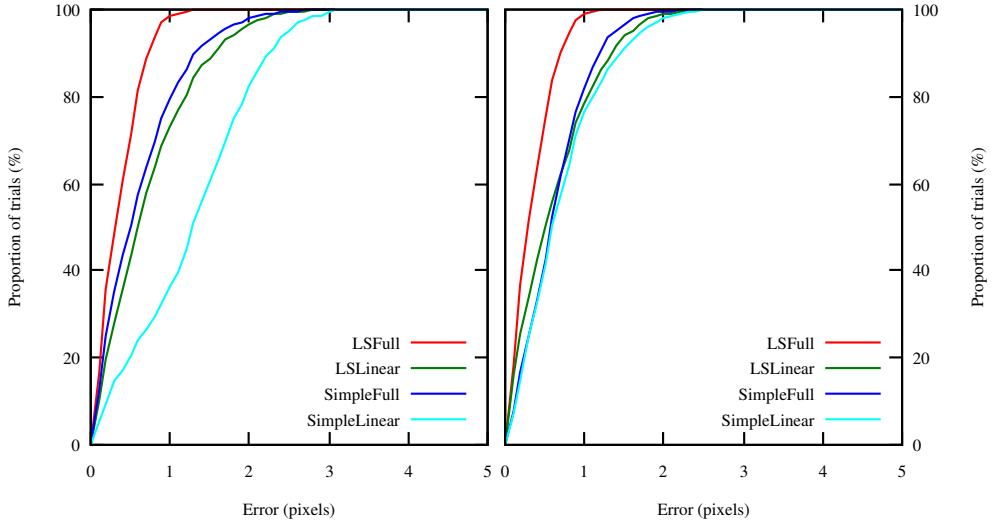
The poor performance of the processing pipelines using the TransformEllipseLinear algorithm may be attributed to approximation error. As shown in Figure 4.20 the TransformEllipseLinear algorithm produces systematic, pathological, displacement of the sample points. This effect is more pronounced in the CircleInner tag because the sample points are extrapolated from the original target bullseye whereas the CircleOuter and CircleSplit tags produce sample points from interpolated values.

4.5.2 Real-world tag reading performance

It is also important to validate that the results from the OpenGL simulation are representative of the system's real-world behaviour. To this end experimental data were collected consisting

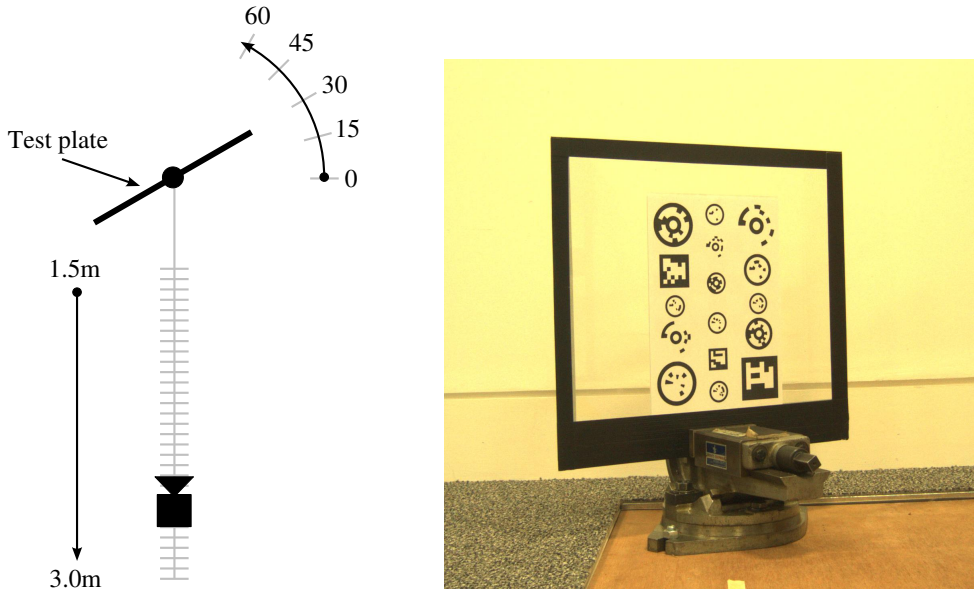


(a) Square (b) CircleInner



(c) CircleSplit (d) CircleOuter

Figure 4.21: Cumulative Error for the Estimated Sample Strength



(a) Experimental Setup (b) Example Image

Figure 4.22: Experimental Setup for real-world testing of Cantag

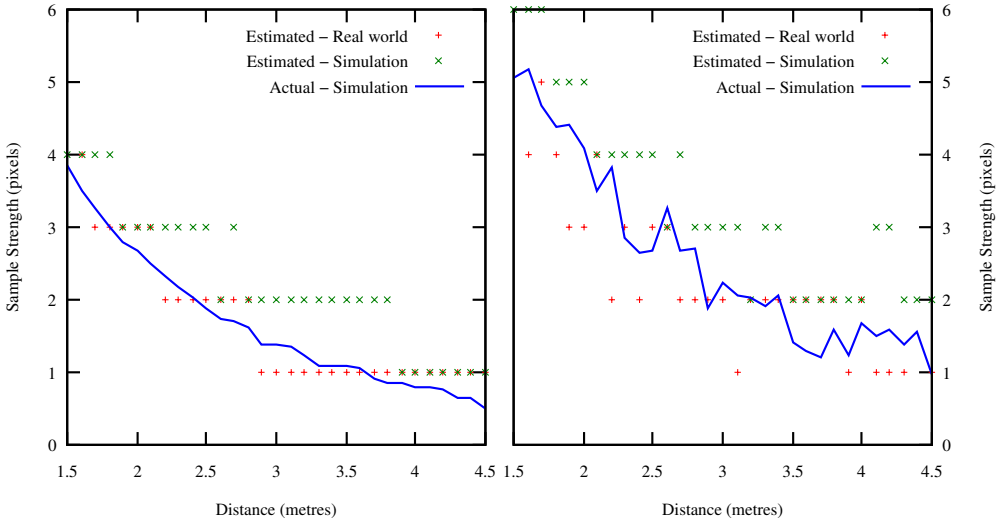
of images at increasing distance and various inclination of a test plate. The experimental setup is shown in Figure 4.22. A digital camera was used for image acquisition with a resolution of 2614×1958 pixels and a field of view of approximately $51°$.

The OpenGL test harness was also configured to simulate a camera with the same intrinsic parameters as used in the experiment. This permits a comparison between outputs from simulated and real-world data.

Figure 4.23 shows that the estimated sample strength values from the real-world data correlate well with the estimated values from the simulated data. Most of the values differ by one pixel or less. These data suggest that the OpenGL simulation is a good model for the real-world behaviour of the system for the current camera in the current operating environment. It can be seen in the figure that for a particular position and pose the Square tag design has a better sample strength than the corresponding circular tag. This is due to the large minimum sample distance of the square tag designs compared to circular designs (Section 4.3). The discrepancy between the estimated sample strength from the simulated data and the actual sample strength is slightly larger for the Square tag than for the CircleSplit tag. This agrees with the cumulative error curves in Figures 4.21(a,c). The estimated sample strength from the real-world images is systematically smaller than the simulated values although the discrepancy is at most one pixel.

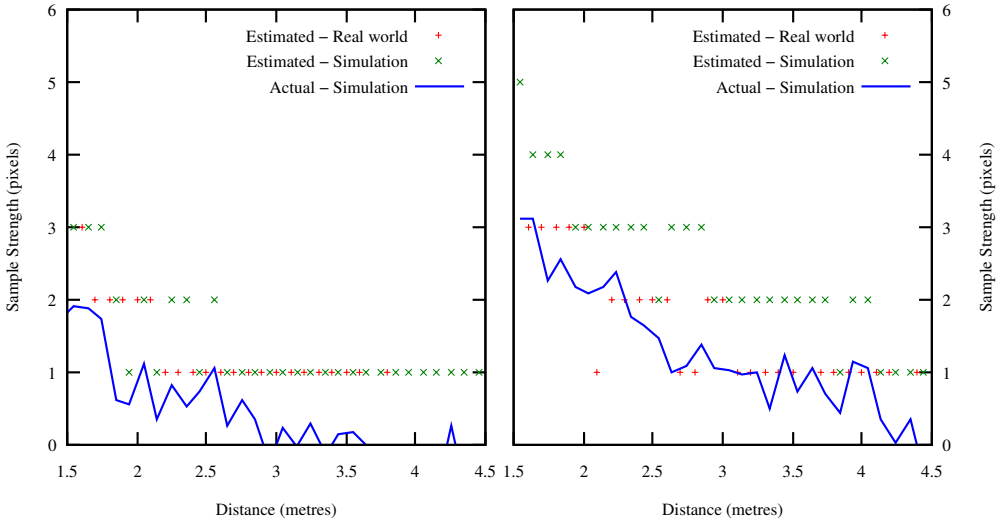
4.5.3 Summary

Sample strength is an important dependability metric: the larger the value the more robust the reading of the current tag data. Before deploying a system designers can make predictions about the sample strength throughout the required tracking space. Positions with large sample strength will be more robust to unmodeled errors in the environment (such as rapidly changing lighting conditions) than those positions with a small sample strength. The estimated sample



(a) CircleSplit at 0 degrees

(b) Square at 0 degrees



(c) CircleSplit at 45 degrees

(d) Square at 45 degrees

Figure 4.23: Sample Strength Estimates using real-world data

strength of a tag reading may be produced without ground-truth knowledge of the tag position. This value may be analysed at runtime to monitor the current performance of the system.

4.6 Location Accuracy

The OpenGL test harness can be used to calculate the error in the estimated tag locations. Figure 4.24 compares the error in location from simulated data with the location errors from the real-world trial. Each sub-figure demonstrates the results from a different processing pipeline or tag design and each column represents a different angle of inclination for the target tag. The sub-figures show the location error (in units of tag size) against the full tag width in pixels—the smaller the tag width the larger the distance from the camera. The location error values shown in each sub-figure have been clamped at 5 tag units so that the trends in the data remain visible despite the noise in the results.

Measurement errors form an inevitable part of the experimental results. The estimated error from the tape measure is 0.1% or 5mm over 5m. This error is not significant compared to errors due to misalignment of the tag plate relative to the camera. This is estimated to be up to 3 degrees which introduces a location error of approximately 40mm (incorporating a deflection both perpendicular and parallel to the tag plate). The tag sizes used are 30mm, 45mm and 60mm which correspond to an estimated systematic error of 1.3, 0.9 and 0.7 tag units respectively for tags at the furthest distance from the camera.

Figures 4.24(a–d) show the performance of the CircleInner tag using the FitEllipseLS algorithm. The error in the simulated data increases as the tag size reduces. This is because the ellipse fitter is able to make a better estimate of the true ellipse if there are more pixels on the contour from which to make the estimate. Increasing the angle of inclination of the tag seems to have little effect because the eccentricity of the ellipse does not affect the capability of the ellipse fitter to fit it. The reduction in the number of points on the ellipse caused by increasing the eccentricity does not seem to be as significant as the reduction caused by reducing the tag size.

In simulation the ideal target bullseye is distorted by pixel truncation in the final image introducing error in the estimated location. Occasionally this truncation will not introduce a significant distortion of the target bullseye and so the system will produce a more accurate result than expected—this effect is evident in the variation in the location accuracy on the graph. A further test of this hypothesis is to run a simulation which begins processing from the image contour which is calculated from the desired position and pose of the tag rather than extracted from a rendered image. In this situation the error remains less than 10^{-3} tag units. The variation in accuracy at this level of precision is most likely ascribed to the accuracy of the floating point number system.

The location errors from the real-world data are similar in shape to those predicted by the simulation. However, the discrepancy is larger than can be explained due to measurement errors in the experiment. This is due to additional real-world effects not modelled in the simulation such as imperfect correction of camera distortion and imperfect thresholding of the original image.

The performance of the FitEllipseSimple algorithm is shown in Figures 4.24(e–h). As expected, the predicted performance of this algorithm is worse than for the FitEllipseLS algorithm. The

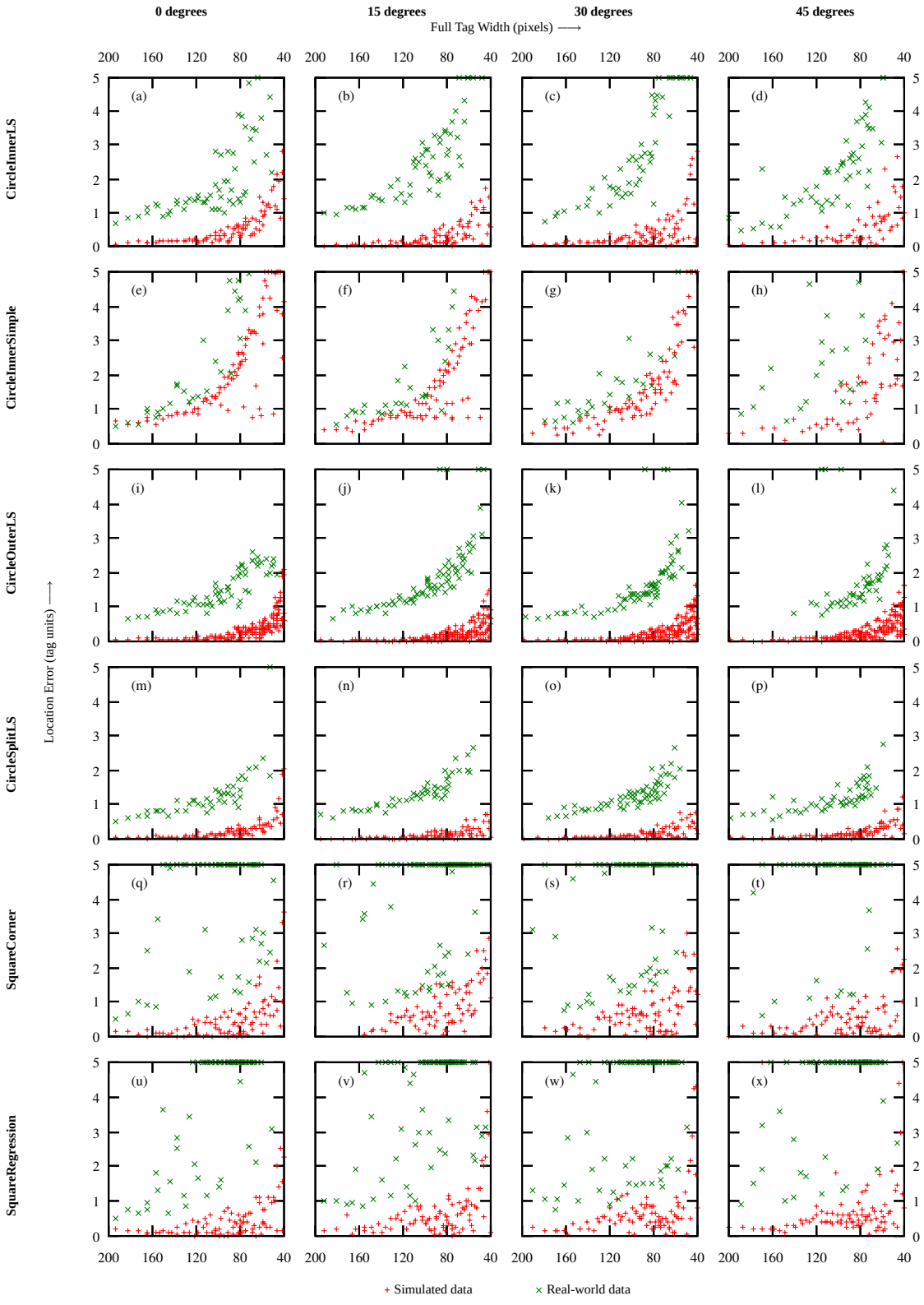


Figure 4.24: Real-world and simulated location error across processing pipelines

relative paucity of sightings derived from this algorithm is due to the fact that poor fitting of the target ellipse results in invalid sampling of the tag code.

The CircleOuter and CircleSplit tags are shown in Figures 4.24(i–l) and (m–p) respectively. These show largely similar performance because the position information is derived from the outer edge of the tag bullseye. Despite the larger radius of the target bullseye, neither design produces significantly more successful readings than the CircleInner design because the data-cells on the tags are now smaller.

The simulated performance for the Square tag design is shown in Figures 4.24(q–x). These pipelines show worse performance than that for the CircleOuter and CircleSplit tags but largely similar to the CircleInner tag. Figures 4.24(q–t) show results using the FitQuadCorner algorithm. This algorithm has poor accuracy because the imaged quadrilateral is estimated from only four points on the image. Adding linear regression to estimate straight lines from the image and placing the tag corners at their intersection increases accuracy slightly; this is shown in Figures 4.24(u–x).

Real-world performance is not close to predicted results. The use of the linear regression technique does not particularly improve matters even though it makes use of the whole contour for estimation. This is because the technique is still dependent upon successful partitioning of the contour into four edge sections prior to the regression. Square tags have the interesting performance characteristic that despite obvious large errors in the recovery of the target shape the data from the tag is still recovered successfully. This is probably due to the fact that incorrect selection of the corners will pull the edges of the square inwards slightly and so the displacement caused to the interpolated sample points is minimal.

The pipeline and tag combinations for the CircleOuter and CircleSplit tags produce data which most closely mirror those produced from simulation. With the current model and analysis these options display more algorithmic dependability than the other pipeline options.

The performance of different vision systems has also been investigated by Zhang et al. [152]. This previous work allowed the identification of the best performing system from the set of those considered. However, it cannot explain why this is the case because of the many factors which differed between the studied systems. The tuneable nature of Cantag allows investigators to decompose these factors and better understand the tradeoffs of individual algorithms.

4.7 Achieving Algorithmic Dependability

Understanding the algorithmic dependability of a location system requires in-depth knowledge and analysis of system operation. This must also be coupled with careful selection of processing algorithms and options to ensure predictable behaviour.

Firstly, the system must be analysed at many levels of abstraction. High level analyses are necessary to gain insight into fundamental behaviour of the tracking technology and mechanism whereas low-level models of system behaviour are required to select promising algorithms and predict their behaviour. For example, the minimum sample distance measure is agnostic to choice of algorithms in the image processing pipeline. This permits an analysis of the fundamental aspects of tag design which is independent of the processing techniques. Similarly,

more refined measures such as sample strength and location accuracy provide insight into the behaviour of particular processing algorithms.

Secondly, simulation of system operation is vital to compare algorithmic performance, evaluate techniques and to locate implementation problems. In the case of estimated sample strength, simulation permitted wide-scale testing of the proposed algorithm. Time limitations and experimental error make these tests implausible on a real-world system. However, some real-world testing must be performed in order to validate the level of realism in the simulation. If real-world performance which significantly differs from that predicted is encountered then either the combination of processing algorithms must be deemed too unstable for dependable operation or further refinement of the model for simulation is required.

Ideally, models should be derived for both predictive and observable analysis. Predictive analysis allows designers to predict the performance of a particular scenario of operation i.e. identifying a marker tag with a chosen position and pose. Observable analysis provides information for applications (and users) utilising data from the system. In this case the true position of the tracked object is not known and so properties must be estimated from the input data rather than calculated from known-truth data. The estimated sample strength algorithm is an example of this. Another example of observable analysis occurs in GPS handsets which estimate the accuracy of a location sighting based on the geometric dilution-of-precision (GDOP) of the satellite constellation used to estimate position.

4.8 Summary

Algorithmic dependability requires understanding the operation of a location system and creating models for its behaviour. This chapter has introduced a number of predictive measures for analysing the performance of a computer vision system. These measures are derived from full knowledge of the tracked object's position and pose. The minimum sample distance expresses the readability of the tag for a given position and pose. This measure has been used as an initial comparison of the expected performance of tag designs and to understand the effects of increasing payload size. Optimal proportions for the three circular tag designs have been derived through the aim of maximising the minimum sample distance.

The concept of sample error reflects the accuracy with which the vision pipeline estimates the sample points for decoding the tag. When combined with the minimum sample distance this yields the sample strength which expresses the tolerance for error in the current tag reading. Deployed systems with large sample strength readings over the active tracking area will be more reliable than systems with small sample strengths.

An algorithm for estimating the sample strength from only the observed image has been presented. The quality of this estimate has been evaluated for a large number of tag positions and poses in simulation and subsequently validated by comparison with real-world data. The estimated sample strength can provide important feedback to a user of the system as to the reliability of the current tag-camera configuration.

The construction of a dependable location system requires particular algorithmic choices in the processing pipeline. CircleSplit and CircleOuter tags utilising the least-squares ellipse fitting algorithm produce real-world location errors close to those predicted by simulation. However,

pipelines built to process square tags produce large errors in the real world not predicted by the processing model. It is unclear whether the analysis and model can be extended to predict these features.

These results suggest that the CircleSplit tag is a good choice for algorithmic dependability whilst maintaining tracking performance. From a dependability point-of-view this design produces little error in the sample strength estimation process and also closely mirrors predicted errors in location accuracy. The design also maintains a larger minimum sample distance (and hence provides better read performance) than the structurally similar CircleOuter tag.

Chapter 5

Runtime Dependability

The theoretical capabilities of a location system, derived from algorithmic dependability analysis, can be a good predictor of real-world performance. However, the metrics which have been developed for location system performance are highly specific to the internals of the location system and must be evaluated at the correct place in the system processing pipeline.

Considerable effort was required to achieve a robust implementation of Cantag (due to both implementation and algorithmic errors). This demonstrates how likely it is that problems and shortcomings will become evident in a real-world deployment which will cause the system to depart from its expected behaviour. Furthermore, due to the requirements of sensing contextual information and the proliferation of mobile devices, Sentient Computing systems are inherently distributed. This exposes the system to additional problems which arise from concurrent execution such as race conditions and distributed failures. Another significant factor in real-world Sentient Computing systems is their dynamic nature. Devices and applications (often carried by users) enter and exit the system on short timescales. This makes it implausible to statically analyse, test, and provision the system.

Runtime validation of a system applies specific, tailored checks to computation blocks to ensure that the results produced are consistent with the input data. This additional implementation diversity aids in catching runtime faults and naturally integrates the evaluation of the metrics arising from algorithmic dependability analysis. Validation enforces transparent operation of a distributed system which in turn ensures that faults can be traced to a particular component of the system.

This chapter applies runtime validation techniques to Sentient Computing infrastructures and demonstrates the dependability benefits which arise. The validation requirements for a system are naturally expressed using inference rules and so a formalism for describing the validation requirements of a Sentient Computing infrastructure is developed. This formalism is subsequently exploited to show the soundness of validation optimisations made to the operation of the original system.

5.1 Runtime Faults in Sentient Computing

Faults can occur in may places and for many reasons during the operation of a Sentient Computing infrastructure.

Distributed operation forms a fundamental underpinning of Sentient Computing. Applications, often running on impoverished devices, are commonly connected over a network to a middleware which provides common support functionality to the entire system. Similarly, wide-scale location systems naturally require far-flung deployments of sensing (and often processing) hardware. This model of computation is inherently fraught and often unreliable. Concurrent, independent execution of the various components of the system often makes failures hard to reproduce or debug.

Location systems perform complex processing on widely varying input data. Occasionally these inputs will combine pathologically and cause faults in the processing algorithms or their implementations. It can be argued that implementation errors should be eliminated by improving software engineering techniques. However, software errors seem to exist in even the most carefully (and expensively) engineered systems. Spaceflight is one of many examples [147] where huge investment is made to achieve reliable software but the smallest mistake has catastrophic results.

Another source of unexpected input values comes from the fact that a location system is designed to measure the environment without overly constraining it. Modelling these inputs becomes arbitrarily complex as more and more detail about the environment is included and so it is plausible that significant unmodelled events can manifest themselves in the input data of the system.

Theoretical errors can also cause faults in the system. For example, when first published, the least-squares ellipse fitting routine used in Cantag contained numerical instabilities. It was later demonstrated that an error-free selection of points lying exactly on the ellipse contour generates an ill-posed set of equations in the matrix operations applied [60].

Checking the output of an algorithm is only effective in this situation if the check is algorithmically independent (rather than just implementation independent) of the original computation.

5.2 Exploiting Asymmetric Computation

Integer multiplication and its inverse, factorisation, is a good example of an asymmetric computation. Multiplication of two prime numbers to form their product is a low-cost operation whereas splitting the result into its two factors is significantly more expensive. The security of the RSA encryption algorithm derives from the relative difficulty of factoring the product of two large, prime numbers.

Analogously, many operations in Sentient Computing also have asymmetric computation costs: the forward computation of the function is time-consuming and computationally intensive, whereas computing the inverse function is cheap and easy.

For example, as described in Section 2.1.8 the Active Bat system executes an iterative, non-linear multi-lateration algorithm for each location sighting. This process uses a non-linear regression to repeatedly hypothesise a position from the set of distance readings from the ultrasonic sensors for a particular Bat sighting. Outlier distances (often due to multi-path signals) are discarded and the process repeats until a precise reading is obtained. Conversely, checking that the output of this algorithm is correct is simply the case of generating a set of expected distance readings and verifying that a suitably sized quorum is consistent with the original distances.

The SPIRIT middleware has facility to provide an application with a callback when pre-defined spatial regions surrounding physical objects interact [3]. Similar functionality is also present in the *IdentityPresence* widget in the Context Toolkit [123]. The continual monitoring for a *containment* event for a large number of arbitrary regions over a large deployment area requires significant computational effort. However, once a trigger event is identified, the verification that the sighting construes the containment event is simple.

A final example, from Cantag, is the ellipse fitting routine. The process of finding the best ellipse fit to a particular contour is expensive to compute and difficult to implement. However, numerous low-complexity techniques exist for checking that an *already fitted* ellipse matches the set of input points [120].

5.2.1 Negative validation

Examples such as containment monitoring and ellipse fitting demonstrate a form of *positive validation*. In these scenarios checking the validity of an output only requires demonstrating that the produced event is consistent with the inputs. More complex, but also common, scenarios require validation that an event has not occurred—so called *negative validation*.

Suppose an application requests that the middleware delivers a callback dependent upon an *entry* event when the user acr31 enters the room SN04. Validation must show that:

- at some time t acr31 is contained in SN04;
- the location of acr31 at the user's previous sighting (before t) at time s places him outside SN04;
- for all times between s and t it is valid that no sighting of acr31 occurred.

This level of checking is required to demonstrate to the application that the event has been raised at the first instance of entry into the region rather than at some later time.

Dependable applications must check that the underlying system is operating correctly at runtime. This can be achieved through the use of the observable metrics developed though algorithmic dependability. These metrics are extremely system specific and often rely on intermediate information within the sensor system. For example, the estimated sample strength metric in Cantag provides an estimate of the tolerance achieved on the current reading of the tag. Execution of this check requires prior knowledge of the tag design in addition to the acquired image and the derived three-dimensional transform for the tag.

A validation framework provides a general means to integrate these tests into the system—they are simply additional application-specific checks to be applied to the relevant stages.

5.3 Expressing Validation Requirements

Validation checks must be executed at runtime in order monitor the correctness of data in the system. These checks might be performed by end-user applications or by autonomous agents deployed within the system.

The specific nature of validation necessitates a different set of checks for each system to be validated. This section defines a logical framework for validation conditions. Within this framework the system is viewed as a set of interconnected processing nodes. The validation of these nodes is expressed as a combination of axioms and inference rules. Expressing a system's validation requirements in this manner is beneficial because it highlights generally applicable reasoning techniques and permits proofs of various properties of the system.

The input and output data of each processing node are assumed to be available as tuples in a tuple-space [57]. Each result, which is often viewed as an event, is assumed to contain a *primary key* which uniquely identifies it. An example primary key might be a frame number allocated from a global clock source composed with a unique identifier for the processing node which produced the event. The following example event denotes that the tag TAG1 was located in frame number 1 by the Data Decoding (DD) node in Cantag (discussed later):

$$(1, \mathrm{DD}, \mathsf{TAG1})$$

The discussion herein treats the node identifier as a *typeid*[1] and thus conceptually assigns different types to events dependent upon the producing node. This commonly reduces the primary key to be isomorphic with some clock source and so it is referred to as the *timeslot* value. Thus, the example above becomes:

$$(1, \mathsf{TAG1})_{\mathrm{DD}}$$

Events in the tuple-space are assumed to be immutable and to persist indefinitely. Practical considerations for the implementation of this are discussed at the end of this chapter (Section 5.9). Validation is defined over a set of logical predicates. The first of these is the *existence* predicate which is used to look-up events in the tuple-space. The predicate $\mathcal{E}_{\mathrm{n}}(t, d)$ is true for a functional unit n if the event data d was emitted at timeslot t. If the example above is in the tuple-space the following predicate holds:

$$\mathcal{E}_{\mathrm{DD}}(1, \mathsf{TAG1})$$

For a particular node n in the system, the two axes of validation (positive and negative) are written as $\mathcal{V}^{+}_{\mathrm{n}}$ and $\mathcal{V}^{-}_{\mathrm{n}}$ respectively. The positive validation predicate $\mathcal{V}^{+}_{\mathrm{n}}$ is true for a particular timeslot if the event for that timeslot exists and is valid. The negative validation predicate $\mathcal{V}^{-}_{\mathrm{n}}$ is true for a particular timeslot if no event was produced and it can be shown it is valid that no event was produced. Due to the difficulty of proving that some event *did not* occur, negative validation is commonly performed by showing that some other event(s) did occur which entail that no event should have been produced by this processing node.

Commonly, the validation of a particular node is predicated upon the validation of its inputs. Given a set of valid inputs it remains to check that the output of the functional unit is consistent with these inputs. This validity checking function is denoted $\mathcal{C}_{\mathrm{n}}(t)$ and is true for some timeslot t if the outputs of the processing node (n) are consistent with its inputs. For example, the check that the DD node produced the correct output at timeslot 1 is expressed as:

$$\mathcal{C}_{\mathrm{DD}}(1)$$

[1] In C++ the typeid operator is used at runtime to discover the type a referenced object from a special type_info object in the referenced object's virtual-function table (vtable) [97, p120]

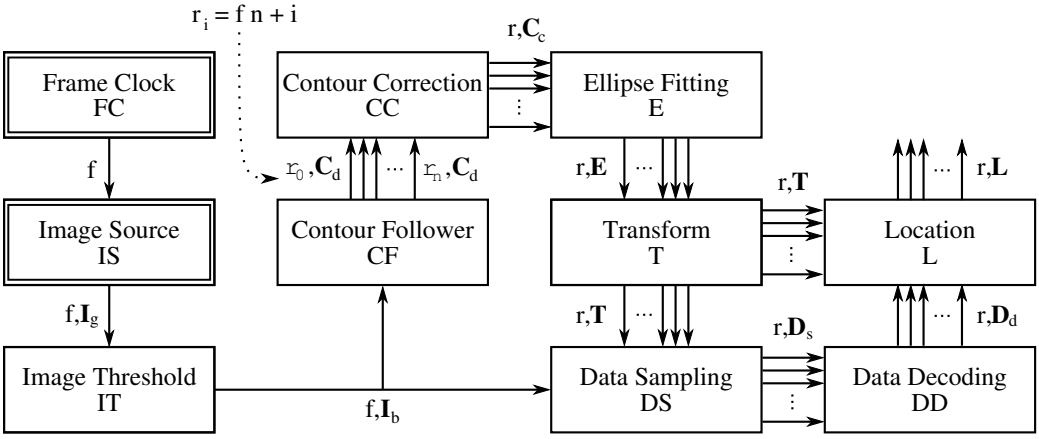


Figure 5.1: Functional diagram of a Cantag processing pipeline. Trusted nodes are shown with a double outline

Multiple checking functions for a particular node are distinguished by a superscript label. For example, $\mathcal{C}^{+}_{\mathrm{DD}}$ and $\mathcal{C}^{-}_{\mathrm{DD}}$ refer to checking functions for the DD node. The first checks a value of a positive result and the second checks that an undefined (negative) result was correct.

It is not possible to validate the behaviour of all nodes in the system. In particular, the output of a node which is reporting data from a sensor cannot be checked because no input exists within the system to validate it against. Nodes for which validation is not possible are referred to as *trusted nodes*; their behaviour is implicitly trusted. In the logical model of system operation the behaviours of these nodes are defined as *axioms* to reflect this. Validation predicates ($\mathcal{V}^{+}_{\mathrm{n}}$ and $\mathcal{V}^{-}_{\mathrm{n}}$) for the remaining (untrusted) processing nodes are specified using *inference rules*.

$$(\mathrm{DD}^{+})\ \frac{\mathcal{V}^{+}_{\mathrm{DS}}(r) \qquad \mathcal{C}^{+}_{\mathrm{DD}}(r)}{\mathcal{V}^{+}_{\mathrm{DD}}(r)}$$

In this example the rule states that the output of the DD node at timeslot r positively validates ($\mathcal{V}^{+}_{\mathrm{DD}}(r)$) if it can be shown that the output of the DS node at timeslot r positively validates ($\mathcal{V}^{+}_{\mathrm{DS}}(r)$) and the output of the DD node at timeslot r is consistent with its inputs ($\mathcal{C}^{+}_{\mathrm{DD}}(r)$).

5.4 A Validation Architecture for Cantag

The validation rules for a particular system are derived from a model of its operation. A functional depiction of the Cantag processing pipeline is shown in Figure 5.1. This pipeline is suitable for tracking circular tags which have already been identified as preferable for their algorithmic dependability advantages over square tags in Chapter 4.

The Contour Follower (CF) node is of particular interest because it splits a single input (the image) into multiple outputs (a set of contours). This is an example of a *divergent node* and must be validated with care. In particular, it must be possible for the validating process to verify that it has collected all of the output data from the node. In the case of the Contour Follower node it must be possible to validate that all contours have been collected as well as

that each of the collected contours is correct. This property, known as *enumeration*, is achieved in the Contour Follower node by including the *raster position* (r) of the start point of the contour in the primary key. The raster position enumerates all pixels in the image one row after another. The raster position is shown in Figure 5.1 for frame number f with n pixels as $r_0, \ldots, r_n$ where $r_i = fn + i$. By iterating over all raster positions the validating application may be sure that all possible events have been checked. If the Contour Follower node had only assigned a unique index to each located contour then there is no means to check that a contours has not been overlooked in the original image without re-running the entire Contour Following process.

The basis of the specification is a number of axioms which define the behaviour of the trusted components of the system:

- For all timeslots (t) the Frame Clock (FC) node generates a clock event ($\top$) and is valid:
$$\forall t.\ \mathcal{E}_{\mathrm{FC}}(t, \top) \wedge \mathcal{V}^{+}_{\mathrm{FC}}(t)$$
- For all timeslots (t) the Image Source (IS) node generates an image and is valid:
$$\forall t.\ \mathcal{E}_{\mathrm{IS}}(t, \mathbf{I}) \wedge \mathcal{V}^{+}_{\mathrm{IS}}(t)$$
- The image generated by the Image Source node is of fixed size and contains n pixels.

The remainder of the system has been specified using inference rules as shown in Figure 5.2. The use of the raster position in the Contour Node is evident in the side condition of the CF rules. In this condition the frame number f is recovered from the raster position r by dividing by the number of pixels in the image r.

Most of the functional units in Cantag implement *total* functions in that they produce an output for every input. Units such as the Contour Follower and Data Decode are *partial* functions because for some inputs (points which do not begin a contour, or invalid tag data) they produce no output or an undefined result. Validation of these nodes requires two checking functions: $\mathcal{C}^{+}$ checks that the result produced by the function is consistent with the input (so called positive validation), and $\mathcal{C}^{-}$ checks that it is valid to have produced an undefined result from the input (so called negative validation).

To validate all events from the system the application must check:

$$\forall r.\ \mathcal{V}^{+}_{\mathrm{L}}(r) \vee \mathcal{V}^{-}_{\mathrm{L}}(r) \tag{5.1}$$

5.5 A Validation Reasoning Engine

The steps required to validate each output of the system are defined by the inference rules for that particular system. The search procedure for selecting which rules to apply is similar to that implemented by logic programming languages such as Prolog. In order to exploit this, each inference rule has been specified as a Horn clause in a Prolog program. The validity of data is determined by posing questions to the reasoning engine.

Figure 5.3 shows the Prolog clauses implementing the validation rules for Cantag. These arise from a straightforward transcription of the axioms for the trusted components and the inference rules in Figure 5.2. The side condition for the CF^{+} and CF^{-} rules is implemented in Prolog using integer division written as `//`.

$$\text{(IT)}\ \frac{\mathcal{V}^+_{\mathrm{IS}}(f) \qquad \mathcal{C}^+_{\mathrm{IT}}(f)}{\mathcal{V}^+_{\mathrm{IT}}(f)}$$

$$\text{(CF}^+\text{)}\ \frac{\mathcal{V}^+_{\mathrm{IT}}(f) \qquad \mathcal{C}^+_{\mathrm{CF}}(r)}{\mathcal{V}^+_{\mathrm{CF}}(r)}\ f = \lfloor r/n \rfloor \qquad\qquad \text{(CF}^-\text{)}\ \frac{\mathcal{V}^+_{\mathrm{IT}}(f) \qquad \mathcal{C}^-_{\mathrm{CF}}(r)}{\mathcal{V}^-_{\mathrm{CF}}(r)}\ f = \lfloor r/n \rfloor$$

$$\text{(CC)}\ \frac{\mathcal{V}^+_{\mathrm{CF}}(r) \qquad \mathcal{C}^+_{\mathrm{CC}}(r)}{\mathcal{V}^+_{\mathrm{CC}}(r)}$$

$$\text{(E}^+\text{)}\ \frac{\mathcal{V}^+_{\mathrm{CC}}(r) \qquad \mathcal{C}^+_{\mathrm{E}}(r)}{\mathcal{V}^+_{\mathrm{E}}(r)} \qquad\qquad \text{(E}^-\text{)}\ \frac{\mathcal{V}^+_{\mathrm{CC}}(r) \qquad \mathcal{C}^-_{\mathrm{E}}(r)}{\mathcal{V}^-_{\mathrm{E}}(r)}$$

$$\text{(T)}\ \frac{\mathcal{V}^+_{\mathrm{E}}(r) \qquad \mathcal{C}^+_{\mathrm{T}}(r)}{\mathcal{V}^+_{\mathrm{T}}(r)}$$

$$\text{(DS)}\ \frac{\mathcal{V}^+_{\mathrm{T}}(r) \qquad \mathcal{V}^+_{\mathrm{IT}}(t) \qquad \mathcal{C}^+_{\mathrm{DS}}(r)}{\mathcal{V}^+_{\mathrm{DS}}(r)}$$

$$\text{(DD}^+\text{)}\ \frac{\mathcal{V}^+_{\mathrm{DS}}(r) \qquad \mathcal{C}^+_{\mathrm{DD}}(r)}{\mathcal{V}^+_{\mathrm{DD}}(r)} \qquad\qquad \text{(DD}^-\text{)}\ \frac{\mathcal{V}^+_{\mathrm{DS}}(r) \qquad \mathcal{C}^-_{\mathrm{DD}}(r)}{\mathcal{V}^-_{\mathrm{DD}}(r)}$$

$$\text{(L}^+\text{)}\ \frac{\mathcal{V}^+_{\mathrm{DD}}(r) \qquad \mathcal{V}^+_{\mathrm{T}}(r) \qquad \mathcal{C}^+_{\mathrm{L}}(r)}{\mathcal{V}^+_{\mathrm{L}}(r)} \qquad \text{(L}^-_1\text{)}\ \frac{\mathcal{V}^-_{\mathrm{CF}}(r)}{\mathcal{V}^-_{\mathrm{L}}(r)} \qquad \text{(L}^-_2\text{)}\ \frac{\mathcal{V}^-_{\mathrm{DD}}(r)}{\mathcal{V}^-_{\mathrm{L}}(r)}$$

Figure 5.2: Inference rules for validation in Cantag

```
vFCp(_).                        % Trusted Component
vISp(_).                        % Trusted Component
n(307200).                      % 640x480 image

vITp(T) :- vISp(T), cITp(T).    % Rule IT
vCFp(S) :- n(R), T is S // R,
           vITp(T), cCFp(S).    % Rule CF+
vCFn(S) :- n(R), T is S // R,
           vITp(T), cCFn(S).    % Rule CF-
vCCp(S) :- vCFp(S), cCCp(S).    % Rule CC
vEp(S)  :- vCCp(S), cEp(S).     % Rule E+
vEn(S)  :- vCCp(S), cEn(S).     % Rule E+
vTp(S)  :- vEp(S), cTp(S).      % Rule T
vDSp(S) :- vTp(S), vITp(S),
           cDSp(S).             % Rule DS
vDDp(S) :- vDSp(S), cDDp(S).    % Rule DD+
vDDn(S) :- vDSp(S), cDDn(S).    % Rule DD-
vLp(S)  :- vDDp(S), vTp(S),
           cLp(S).              % Rule L+
vLn1(S) :- vCFn(S).             % Rule L1-
vLn2(S) :- vDDn(S).             % Rule L2-
vLn(S)  :- vLn1(S) ; vLn2(S).
```

Figure 5.3: Validation rules for Cantag (Prolog clauses)

```
checkAll(-1).
checkAll(T) :- ( vLn(T) ; vLp(T) ),
               S is T-1, !, checkAll(S).
```

Figure 5.4: System validation (Prolog clauses)

5.5.1 Implementing the check predicates

The values of the check predicates ($\mathcal{C}^+_{\mathrm{IT}}, \mathcal{C}^+_{\mathrm{CF}}, \ldots$) depend on the current state of the system requiring validation. Their implementation is efficiently achieved using the Prolog foreign language interface such that, for example, the act of querying the predicate $\mathcal{C}^+_{\mathrm{E}}(r)$ may actually call the C++ code within Cantag to check whether the ellipse at raster position r is correctly derived from its input contour.

The check predicates must be evaluated during the search through the inference rules rather than after the search because their value informs the reasoning engine if current path is valid. In Cantag, for example, the `vLn` clause can be satisfied if either `vLn1` or `vLn2` are true—the choice of which is suitable depends on the values of the check predicates.

Validating all data from the system requires an implementation of Equation 5.1. The Prolog clauses for this are shown in Figure 5.4. This formulation departs from the logical form because it is necessary to direct the search path for the Prolog inference engine to cover all raster positions for all frames. However, the intent of the original equation is still clear in that one of the validation rules for the Location node must be true and this must also be the case for all previous raster positions. The addition of the cut operator (`!`) causes the reasoning engine to discard all choice points in the search space and commit to the current path. This therefore reduces the memory cost of evaluating the search because all the state required for back-tracking from the current point to attempt a different path can be discarded. This is analagous to the difference between recursive and tail-recursive functions in functional programming.

5.5.2 The validation process and costs

The foreign language predicate calls have been implemented with the ability to output debugging information whenever they are invoked. This allows visualisation of the validation trace. A test environment was constructed wherein all positive check functions ($\mathcal{C}^+_{\mathrm{n}}$) are true for all timeslots and all negative check functions ($\mathcal{C}^-_{\mathrm{n}}$) are always false. The output created by the `checkAll` predicate is presented in Figure 5.5. The output has been reformatted and cropped for clarity.

The query shown in the figure activates the debug output of the checking predicates and requests that the system check the validity beginning with the tenth raster position of the first frame (frame zero, raster position 10).

The first block of traced calls shows the reasoner's first attempt by checking `vLn(10)`. This triggers a test for `vCFn(10)` which fails (after recursing to check `vITp(0)`) because the check `cCFn(10)` returns false. Calling `vITp(0)` is necessary to check the thresholding output for the first frame—the primary key here is the frame number.

```
?:- enableDebug(1), checkAll(10).

cITp(0) cCFn(10)

cITp(0) cCFp(10) cCCp(10) cEp(10) cTp(10)
cITp(10) cDSp(10) cDDn(10)

cITp(0) cCFp(10) cCCp(10) cEp(10) cTp(10)
cITp(10) cDSp(10) cDDp(10) cITp(0) cCFp(10)
cCCp(10) cEp(10) cTp(10) cLp(10)

cITp(0) cCFn(9)

...

cCCp(0) cEp(0) cTp(0) cLp(0)
Yes
```

Figure 5.5: An example validation session for Cantag

The second block shows the next attempt by using the second available option for showing `vLn(10)` by checking `vDDn(10)`. This also fails.

In the third block the reasoner attempts to show `vLp(10)` which succeeds. This enables the reasoner to recurse and attempt to show `vLn(9)` and so on until the first raster position is reached and validated (`vLp(0)`).

It can be seen from this trace that the same predicates are repeatedly called as the search continues. In fact the trace contains 264 calls to checking predicates but only 110 of these are distinct. It is therefore important to cache the results of expensive checking functions. This can be done entirely in the native implementation or by up-calling the Prolog engine from the native code to insert new ground clauses into the database.

Figure 5.6 shows the number of distinct calls required in a real run of the system. The image sequence used showed three tags entering and exiting the field-of-view. Each graph shows the number of calls to the checking function against the frame number. This makes n (the maximum raster position) the largest possible number of calls to each checking function per frame. In actuality the vertical scale on all the graphs except $\mathcal{C}^{-}_{\mathrm{CF}}$ is from zero to 140. The majority of raster positions do not correspond to a contour and so $\mathcal{C}^{-}_{\mathrm{CF}}$ is called hugely more often than any other check.

5.5.3 Improving performance

The computational cost of evaluating a particular checking function is problematic to measure because the cost is highly dependent upon the particular implementation of the function, the situation it has been executed in and the machine executing the code. However, it is possible to make algorithmic comparisons between the cost of computing the checking function and the

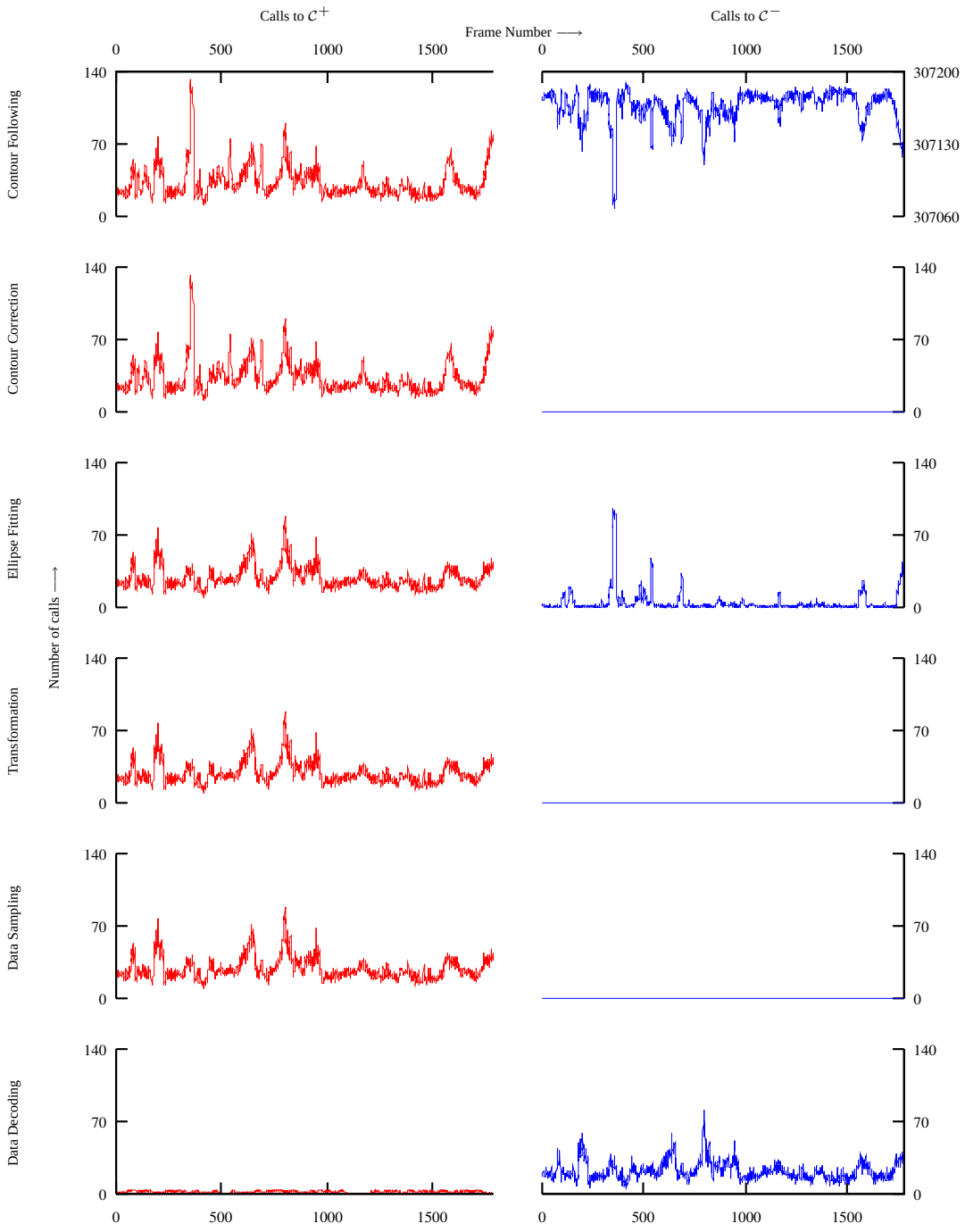


Figure 5.6: The number of calls to positive and negative checking functions

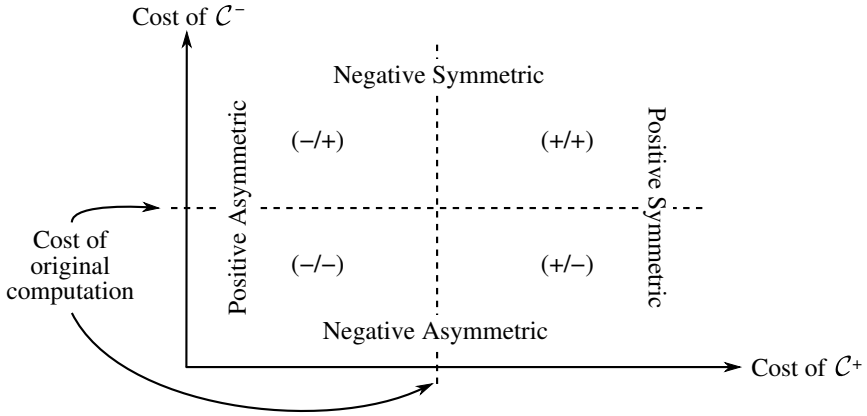


Figure 5.7: Classification of the computation costs of the checking function

cost of the original computation that produced the result. Total functions (those which produce an answer for all input values) may be described as *symmetric* or *asymmetric*. The cost of checking the output of symmetric functions is comparable to, or greater than, the cost of the original computation. For asymmetric functions, the cost of checking is less than that of the original computation. The classification for partial functions (which return an undefined value for some inputs) is shown in Figure 5.7. The following sections explain the four classes in detail.

Positive symmetric functions

A positive symmetric function has a checking function which takes comparable or greater time than the original computation when checking that produced data are valid. An example is the Contour Follower node for which checking that a contour is correct for a given raster position requires following the contour around the image—the same cost as originally required to locate it initially.

Positive asymmetric functions

For this class of functions it is cheaper to check positive outputs than it is to compute them in the first place. The Transformation node in Cantag is an example of this. Deriving the three-dimensional position and pose of the tag from its image is a complex operation. However, given a proposed transformation it is relatively cheap to project a number of test points and check that they correlate with the image.

Negative asymmetric functions

Negative asymmetric functions require less computational effort to check an undefined output is valid than the original computation of the undefined value. In general, if it were possible to write a checking function for discarded input data which is cheaper than the original computation which discarded the input data then the checking function could be incorporated into the original

computation to improve performance. This makes the existence of pure negative asymmetric functions marginal.

Some form of this behaviour can arise if an algorithm is altered to output some diagnostic with failed input data. The check functions may now use this diagnostic to cheaply check validity. However, by the pure definition of total and partial functions, this function would no longer be partial because it now produces some output for all inputs.

Negative symmetric functions

Negative symmetric functions take similar, or greater time, than the original computation to check an undefined result. For the Contour Follower node, checking that a particular raster position does not correspond to a start point of a contour falls into one of two scenarios: if the point under the raster position does not correspond to an edge in the image then no contour could begin here; if the point lies on an edge the check function must check that the point cannot be a start point of the contour. This is achieved by traversing the contour searching for a point with a lower raster position than the candidate point. Locating a lower raster position point on the same contour is sufficient to demonstrate that the candidate point cannot be the start point of the contour.

Many nodes such as the Ellipse Fitter or the Data Decoding node have this behaviour because the technique for checking that an undefined value is valid is to re-run the original algorithm (or another implementation of it). This is not ideal from a validation point-of-view because this check re-uses the same or a similar code path and so is more likely to expose the system to theoretical errors in the algorithm.

This problem can be alleviated by attempting to reject invalid inputs at easier-to-validate nodes earlier in the pipeline. For example, if the data for a particular raster position can be excluded from consideration then all future stages of the pipeline benefit because it is no longer necessary to check the validity of that position. This reduces the number of discarded entities at the Ellipse Fitting stage (say) and so reduces the impact of the weaker validation properties of the node.

The minimum sample distance measure presented in the previous chapter (Section 4.3) specifies a minimum tag size in pixels, below which it is not theoretically possible to recover the data payload. Applying a heuristic that discards contours at the Contour Follower node which fall below this minimum size reduces the number of raster points which propagate through the pipeline. This heuristic can be expressed as an additional inference rule:

$$(\mathrm{CF}_2^-)\ \frac{\mathcal{V}_{\mathrm{IT}}^+(t) \qquad \mathcal{C}_{\mathrm{SIZE}}^-(r)}{\mathcal{V}_{\mathrm{CF}}^-(r)}\ t = \lfloor r/n \rfloor$$

Figure 5.8 shows the effect of this new rule on the number of calls to the checking functions. Each of the graphs (including $\mathcal{C}_{\mathrm{SIZE}}^-$) has a vertical scale from zero to 140 (as previously in Figure 5.6). The number of remaining raster positions in each frame is significantly reduced. Also, the number of calls to the negative validation of the Ellipse Fitting node is now almost zero.

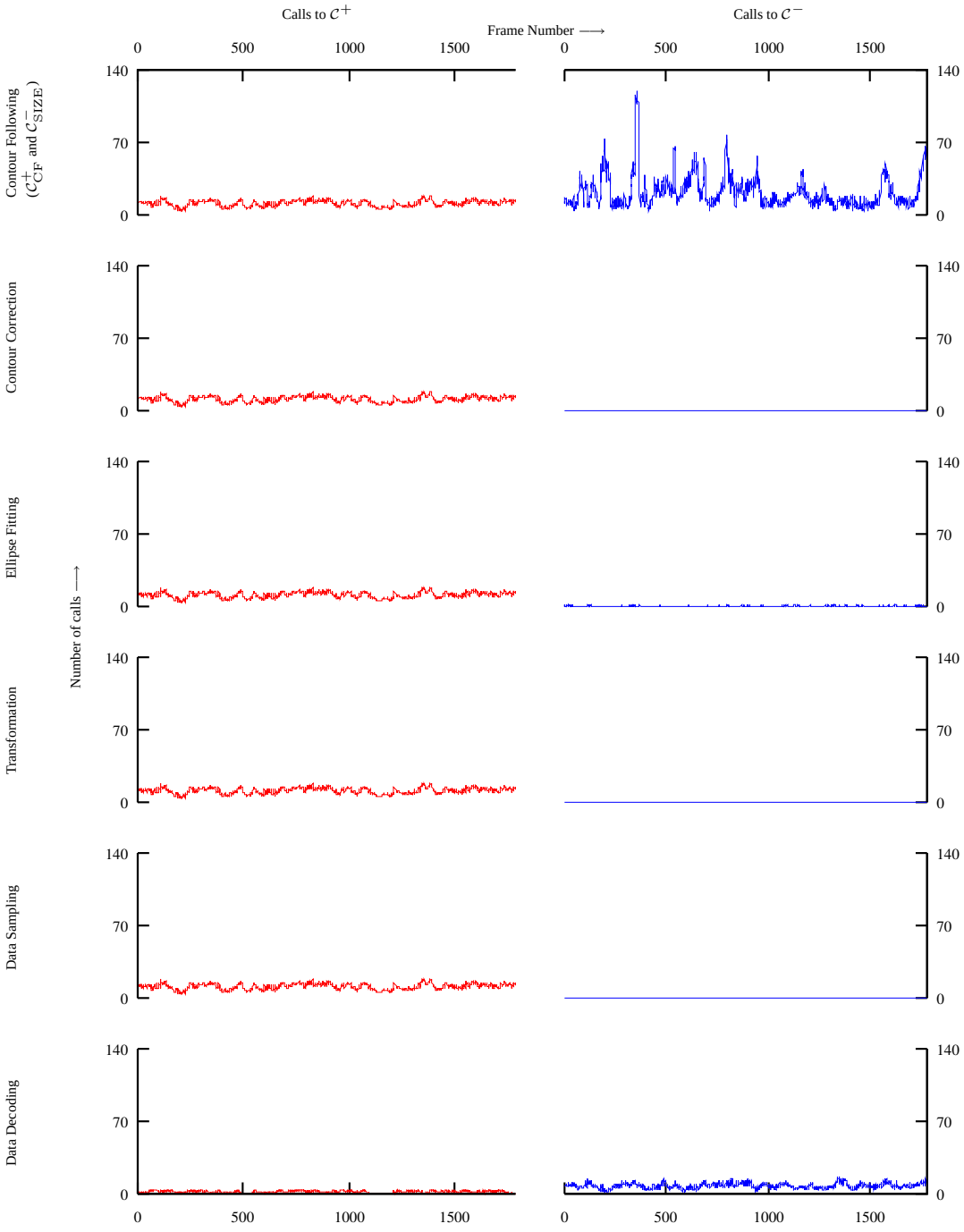


Figure 5.8: Early rejection of candidate entities reduces validation costs

The effect of this (or any other heuristic optimisation) on arbitrary data cannot be predicted from this single example. However, the contour size heuristic can be expected to perform well. Firstly, due to its grounding in the algorithmic limits of the system, applications can be sure that it is not causing the system to generate false negative sightings. Also, in terms of performance, a pathological input containing many contours all slightly larger than the minimum size will cause the system to perform no worse than if the heuristic were not present at all.

Prolog provides a reasoning engine which applies well to the formal inference rules used to specify validation. Designers wishing to support validation need to provide two facilities:

- **Prolog clauses**: these are derived from the functional description of the system and provides information for applications to reason about a system's validity.
- **Checking functions**: the Prolog environment must be extended with native predicates for checking specific information in the system requiring validation.

The judicious addition of new rules can be used to reduce the computational cost of validation. However, in general, applications are not interested in every event generated by the location system and so should not incur validation costs for irrelevant data. The next section extends the logical framework with *selection predicates* which allow applications to discriminate between events and hence further reduce validation costs.

5.6 Application-Oriented Validation

Applications are typically interested in only a subset of events generated by the location system. In this case there is no need to incur the computation cost of validating these unused events. Identifying which events require validation has an additional benefit because it allows applications to ignore a system failure if it does not affect the required context. Ignoring irrelevant system failure is a desirable property for improving system availability. In general, it is safe for an application to do this when the system has failed (and does not validate), if no pertinent information would have been produced from the system *even if it had been working*.

An application's interest in a particular piece of context is modelled by the use of a *selection* predicate. The selection predicate evaluates to a true value for a particular timeslot if the event for that timeslot is of interest to the application. This is a formal expression of the particular contextual information required by the application.

An example is the CONT selection predicate which tracks the tag with identifier $\mathsf{TAG1}$ in a particular region of space. Given an inclusion function $C(x, y, z)$ which is true if the point x, y, z lies inside the container of interest, the selection predicate may be specified as:

$$\mathrm{CONT}(r) \equiv \mathcal{E}_{\mathrm{L}}(r, \{x, y, z, \mathsf{TAG1}\}) \wedge C(x, y, z)$$

The positive validation rule for this predicate is therefore:

$$(\mathrm{CONT}^{+})\ \frac{\mathcal{V}^{+}_{\mathrm{L}}(r) \qquad \mathcal{E}_{\mathrm{L}}(r, \{x, y, z, \mathsf{TAG1}\}) \qquad C(x, y, z)}{\mathcal{V}^{+}_{\mathrm{CONT}}(r)}$$

$$(\mathrm{CONT}_1^-)\ \frac{\mathcal{V}^-_{\mathrm{CF}}(r)}{\mathcal{V}^-_{\mathrm{CONT}}(r)}$$

$$(\mathrm{CONT}_2^-)\ \frac{\mathcal{V}^+_{\mathrm{D}}(r) \qquad \mathcal{E}_{\mathrm{D}}(r, \mathbf{D}) \qquad \mathbf{D} \neq \mathsf{TAG1}}{\mathcal{V}^-_{\mathrm{CONT}}(r)}$$

$$(\mathrm{CONT}_3^-)\ \frac{\mathcal{V}^+_{\mathrm{L}}(r) \qquad \mathcal{E}_{\mathrm{L}}(r, \{x, y, z, \mathsf{TAG1}\}) \qquad \neg C(x, y, z)}{\mathcal{V}^-_{\mathrm{CONT}}(r)}$$

Figure 5.9: Negative validation rules for the CONT selection predicate

The negative validation rules must show that the selection predicate is not satisfied. They should be specified with the minimum possible validation requirements. These are shown in Figure 5.9.

The reason for specifying the minimum requirements for negative validation is to permit the application to ignore a validation failure if it is not relevant. In this example, if the application can show that the identifier on the tag is not of interest then it does not matter which location was recovered for the tag. This results in minimisation of the number of validation steps which benefits the application both in terms of reduced costs for validation and in reducing the likelihood that a system failure will affect it.

Further refinement of validation for the CONT predicate is possible by considering the projection of the region of interest onto the camera image. Any located tag with a contour which starts outside this projected area cannot possibly be later resolved to be inside the region. It is possible to define a predicate ($C_2(r)$) which is true for those primary keys which correspond to contours which begin inside the projected area of the region. This predicate is defined in a straightforward manner by examining the primary key (r) and recovering the raster position ($r \bmod n$). This allows the addition of a further negative validation rule:

$$(\mathrm{CONT}_4^-)\ \frac{\mathcal{V}^+_{\mathrm{CF}}(r) \qquad \neg C_2(r)}{\mathcal{V}^-_{\mathrm{CONT}}(r)}$$

This new rule permits a large reduction in the number of events which require validation because candidate contours may be discarded much earlier in the validation process.

Selection predicates can be used to model the application's contextual requirements. The specification of these should be accompanied by the validation requirements for the predicate. Designers should ensure that negative validation of the predicate has minimal requirements of validity from the rest of the system in order to permit applications to ignore irrelevant failures. The example of the CONT predicate shows how minimal validation requirements can be achieved by expanding the selection predicate constraints back through the system in order to generate rules for discarding false candidates at an early stage.

$$(\text{LAST}_1)\ \frac{\mathcal{V}^+_{\text{p}}(u)}{\text{LAST}_{\text{p}}(u,u)} \qquad (\text{LAST}_2)\ \frac{\text{LAST}_{\text{p}}(u-1,v) \qquad \mathcal{V}^-_{\text{p}}(v)}{\text{LAST}_{\text{p}}(u,v)}$$

Figure 5.10: Inference rules for LAST_p

5.6.1 The LAST predicate

A location system measures the current state of the world. However, the vast majority of applications using location information are event-based. From an application's point-of-view this provides efficient operation because data are only sent to the application for relevant changes in the world state and so network usage is minimal. Event-based operation also implicitly off-loads the calculations required to observe the state change into the middleware, further lowering the application's resource requirements. A validation framework must therefore allow applications to validate system events as well as system state.

The function $\text{LAST}_p(u, v)$ is parameterised over a selection predicate p and is true if, and only if, it can be validated that at time u, v is the most recent time prior to u that p was true. The inference rules for LAST_p are given in Figure 5.10: for the current time u, an application wishing to validate that the last time the selection predicate CONT was true is timeslot v must show $\text{LAST}_{\text{CONT}}(u, v)$ is true. This in turn requires that $\mathcal{V}^+_{\text{CONT}}(u)$ and $\mathcal{V}^-_{\text{CONT}}(t)$ is true for all t after v up to, and including, timeslot u.

Any event can be viewed as a transition between two system states. The validation of an event at some time t therefore requires validation of the state which was entered at t and the validation that the state exited was the last state prior to t. Thus, the LAST predicate forms the basis for the validation of all other temporally-based events.

5.6.2 Entry events

A further example using the LAST predicate is the *entry* event. First, it is necessary to define an additional selection predicate EXCL which is true if the subject is outside (excluded from) the region defined by C:

$$\text{EXCL}(r) \equiv \mathcal{E}_{\text{L}}(r, \{x, y, z, \mathsf{TAG1}\}) \wedge \neg C(x, y, z)$$

This permits the expression of an entry event into the region defined by C at time t as:

$$\exists u.\ \text{LAST}_{\text{EXCL}}(u, t-1) \wedge \mathcal{V}^+_{\text{CONT}}(t)$$

The positive and negative validation rules for CONT and EXCL can be expressed as Prolog clauses in a similar manner to previous rules. This is shown in Figure 5.11. The negative validation rules for both predicates are ordered to try the simplest negative validation rule first before progressing to the more complex checks.

Figure 5.12 shows the implementation of the LAST predicate. This implementation departs from the logical specification of the predicate in order to terminate the search if the engine attempts to validate the timeslot -1. This occurs, for example, if the user attempts to validate

```
r(X,Y,Z)  :- 0 =< X, X =< 5,
             0 =< Y, Y =< 5,
             0 =< Z, Z =< 5.
r2(R)     :- n(N), R mod N =:= 3.

cont(T)   :- eL(T,[X,Y,Z,target]), r(X,Y,Z).

vContP(T) :- vLp(T), eL(T,[X,Y,Z,target]), r(X,Y,Z).
vContN(T) :- vCFn(T) ;
             vCFp(T), \+r2(T) ;
             vDDp(T), eDD(T,D), D \== target ;
             vLp(T), eL(t,[X,Y,Z,target]), \+r(X,Y,Z).


excl(T)   :- eL(T,[X,Y,Z,target]), \+r(X,Y,Z).

vExclP(T) :- vLp(T), eL(T,[X,Y,Z,target]), \+r(X,Y,Z).
vExclN(T) :- vCFn(T) ;
             vCFp(T), r2(T) ;
             vDDp(T), eDD(T,D), D \== target ;
             vLp(T), eL(t,[X,Y,Z,target]), r(X,Y,Z).
```

Figure 5.11: Validation of CONT and EXCL (Prolog clauses)

```
last(_,_,-1,_)          :- !, fail.
last(VPP,_,TLAST,TLAST) :- call(VPP,TLAST).
last(VPP,VPM,T,TLAST)   :- call(VPM,T), !,
                           S is T-1,
                           last(VPP,VPM,S,TLAST).
```

Figure 5.12: Implementation of LAST (Prolog clauses)

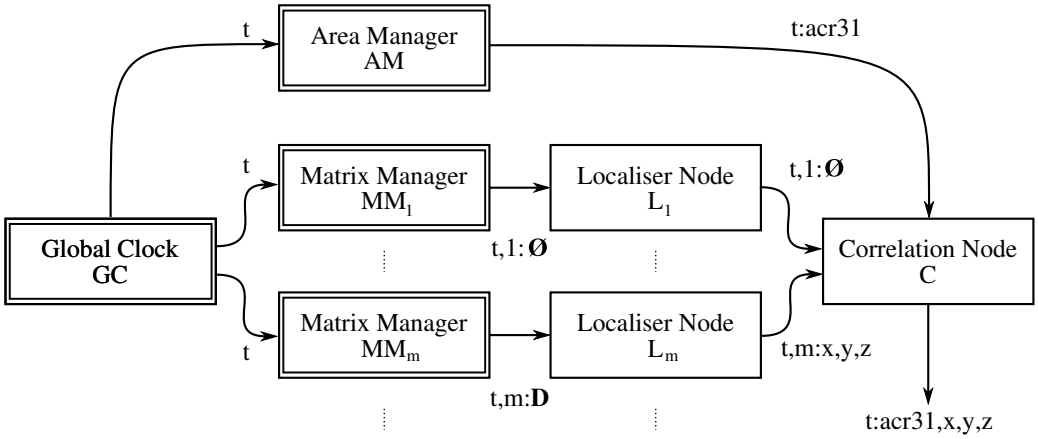


Figure 5.13: Functional diagram of the Active Bat system. Trusted nodes are shown with a double outline

an event which has never occurred. The use of the cut operator in the recursive clause discards the existing search space in order to save stack space (similarly to the implementation of `checkAll`).

Evaluation of the LAST predicate is potentially very expensive because it requires negative validation of the selection predicate for each preceding timeslot until the selection predicate is satisfied. However, in some cases, considerable improvement can be made to this. An example of this occurs in the Active Bat system wherein the polled nature of the system permits the application to make stronger assumptions about which events require validation.

5.7 Validation for the Active Bat system

The basic operation of the Active Bat system is shown in Figure 5.13. The presented operation is a functional model of system operation rather than a true architectural description although terminology is re-used where suitable. The *area-manager* (AM) is responsible for scheduling and selects up to one Bat per timeslot for polling. The Bat is polled over the radio interface and in turn broadcasts a narrow-band "squeak" over the ultrasound channel. Simultaneously, the polling action triggers a timer reset for the ultrasonic location sensors deployed in the ceiling array—typically there is one array per room. Upon receipt of the ultrasonic pulse the ultrasonic sensors relay the estimated time-of-flight to the room's *matrix manager* (MM). The multi-lateration algorithm is subsequently executed upon the set of time-of-flight readings in the room's *localizer node* (L) to produce a location reading for the timeslot. Finally, the *correlation node* (C) pairs the produced location reading with the identifier of the polled Bat based on the timeslot number. Timeslots in the system are allocated by the *global clock* (GC) which is assumed to be available throughout the system. A system deployment of the Active Bat system includes one or more area-managers each of which can be considered to be allocated its own portion (colour) of the available timeslots. This discussion considers a system with a single area-manager; generalisation to many area-managers is straightforward.

The system operates with a set of identifiers for Bats (**I**) extended with $\perp$. The AM emits identifier $\perp$ to indicate no Bat has been polled. This model of the Bat system requires that the

$$(\mathrm{L}^{+}_{m})\ \frac{\mathcal{V}^{+}_{\mathrm{MM}_m}(t) \qquad \mathcal{C}^{+}_{\mathrm{L}_m}(t)}{\mathcal{V}^{+}_{\mathrm{L}_m}(t)} \qquad (\mathrm{L}^{-}_{m,1})\ \frac{\mathcal{V}^{-}_{\mathrm{MM}_m}(t)}{\mathcal{V}^{-}_{\mathrm{L}_m}(t)} \qquad (\mathrm{L}^{-}_{m,2})\ \frac{\mathcal{V}^{+}_{\mathrm{MM}_m}(t) \qquad \mathcal{C}^{-}_{\mathrm{L}_m}(t)}{\mathcal{V}^{-}_{\mathrm{L}_m}(t)}$$

$$(\mathrm{ONE}_1)\ \frac{\mathcal{V}^{+}_{\mathrm{L}_n}(t) \qquad \mathcal{V}_{\mathrm{NONE}}(t, n-1)}{\mathcal{V}_{\mathrm{ONE}}(t, n)} \qquad (\mathrm{ONE}_2)\ \frac{\mathcal{V}^{-}_{\mathrm{L}_n}(t) \qquad \mathcal{V}_{\mathrm{ONE}}(t, n-1)}{\mathcal{V}_{\mathrm{ONE}}(t, n)}$$

$$(\mathrm{ONE}_3)\ \frac{\mathcal{V}^{-}_{\mathrm{L}_n}(t) \qquad \mathcal{V}_{\mathrm{NONE}}(t, n-1)}{\mathcal{V}_{\mathrm{NONE}}(t, n)} \qquad (\mathrm{ONE}_4)\ \frac{}{\mathcal{V}_{\mathrm{NONE}}(t, -1)}$$

$$(\mathrm{C}^{+})\ \frac{\mathcal{V}^{+}_{\mathrm{AM}}(t) \qquad \mathcal{V}_{\mathrm{ONE}}(t, m) \qquad \mathcal{C}^{+}_{\mathrm{C}}(t)}{\mathcal{V}^{+}_{\mathrm{C}}(t)}$$

$$(\mathrm{C}^{-}_{1})\ \frac{\mathcal{V}_{\mathrm{NONE}}(t, m)}{\mathcal{V}^{-}_{\mathrm{C}}(t)} \qquad (\mathrm{C}^{-}_{2})\ \frac{\mathcal{V}^{+}_{\mathrm{L}_p}(t) \qquad \mathcal{V}^{+}_{\mathrm{L}_q}(t)}{\mathcal{V}^{-}_{\mathrm{C}}(t)}\ p \neq q$$

Figure 5.14: Inference rules for validation in the Active Bat System

outputs of the GC, AM, and MM nodes are implicitly trusted. The behaviour of the trusted nodes is specified as follows:

- At each timeslot t the GC node generates a clock tick ($\top$) and is valid:

$$\forall t.\ \mathcal{E}_{\mathrm{GC}}(t, \top) \wedge \mathcal{V}^{+}_{\mathrm{GC}}(t) \tag{5.2}$$

- At each timeslot t The AM node polls one Bat (from the set $\mathbf{I}$) or $\perp$ and is valid:

$$\forall t.\ \exists i \in \mathbf{I} \cup \{\perp\}.\mathcal{E}_{\mathrm{AM}}(t, i) \wedge \mathcal{V}^{+}_{\mathrm{AM}}(t) \tag{5.3}$$

- At each timeslot, every MM is either positively or negatively valid. The MM node might not produce a sighting, in these instances it is assumed that this is valid behaviour:

$$\forall tm.\ \mathcal{V}^{+}_{\mathrm{MM}_m}(t) \vee \mathcal{V}^{-}_{\mathrm{MM}_m}(t) \tag{5.4}$$

Figure 5.14 depicts inference rules for validating the output of each functional unit of the system. Both the Localizer and Correlation nodes are partial functions. The Localizer node produces an undefined output if too few distance readings converge on a unique location (checked by $\mathcal{C}_{\mathrm{L}_m}(t)$) or if the Matrix Manager does not produce an output. The Correlation node is only valid if exactly one Localiser node produces a sighting for the timeslot. This check is implemented using the $\mathcal{V}_{\mathrm{ONE}}$ predicate: either the current Localizer is positively valid and all of the remaining Localizers are negatively valid (as checked by $\mathcal{V}_{\mathrm{NONE}}$); or the current Localizer is negatively valid and $\mathcal{V}_{\mathrm{ONE}}$ is true of the remaining Localizers. Similarly, negative validation of the Correlation node succeeds if none, or more than one, of the Localiser nodes produces a sighting.

Applications wishing to monitor that the entire system is functioning correctly for every sighting must check that, for all timeslots, the correlation node is either positively valid (the location of the Bat validates) or negatively valid (it is valid that no location was produced):

$$(\mathrm{ID}^{+})\ \frac{\mathcal{V}^{+}_{\mathrm{C}}(t) \qquad \mathcal{E}_{\mathrm{C}}(t, \{\mathrm{acr31}, \mathbf{L}\})}{\mathcal{V}^{+}_{\mathrm{ID}}(t)}$$

$$(\mathrm{ID}^{-}_{1})\ \frac{\mathcal{V}^{+}_{\mathrm{AM}}(t) \qquad \mathcal{E}_{\mathrm{AM}}(t, i)}{\mathcal{V}^{-}_{\mathrm{ID}}(t)}\ i \neq \mathrm{acr31} \qquad (\mathrm{ID}^{-}_{2})\ \frac{\mathcal{V}^{-}_{\mathrm{C}}(t)}{\mathcal{V}^{-}_{\mathrm{ID}}(t)}$$

Figure 5.15: Validation inference rules for the ID predicate

$$\forall t.\ \mathcal{V}^{+}_{\mathrm{C}}(t) \vee \mathcal{V}^{-}_{\mathrm{C}}(t)$$

The following discussion considers a more discriminating application which tracks events for the user acr31. Events pertaining to other users are not directly used by the application but they cannot be completely ignored. Each of these extraneous events must be checked to be sure that it is valid that it does not pertain to the user acr31. It is of interest how the application may efficiently perform this check.

Events of direct interest to the application are specified using a selection predicate:

$$\mathrm{ID}(t) \equiv \exists l.\ \mathcal{E}_{\mathrm{C}}(t, \{\mathsf{acr31}, l\})$$

This predicate is true for a timeslot t if the user acr31 has been located at some location l. Figure 5.15 shows the requirements for positive ($\mathcal{V}^{+}_{\mathrm{ID}}$) and negative ($\mathcal{V}^{-}_{\mathrm{ID}}$) validation of this predicate. Notably, for negative validation it suffices to show that the AM node is valid and that the Bat polled does not have identity acr31.

5.7.1 Improving performance for the LAST predicate

The ID selection predicate combines with the LAST predicate (defined in Section 5.6.1) in the obvious way. This combination allows applications to demonstrate the last known position of acr31. Checks for an *entry* can be constructed in an analagous manner to that shown for Cantag (Section 5.6.2).

The cost of checking $\mathrm{LAST}_{\mathrm{p}}(u, v)$ is of the order of the number of timeslots between v and u. Validation over a long time period will require a huge number of checks. However, it is possible to substantially reduce this cost by adding additional functionality to the original system. This change permits low-cost validation of a search predicate which in turn obliviates the need to validate all the intervening timeslots between v and u.

It is first assumed that there exists a search predicate $\mathcal{S}_p(n, t)$ defined for a selection predicate p. The search predicate gives the timeslot t such that the predicate p becomes true for the nth time. The actual implementation of this predicate is discussed shortly. The properties of this predicate are defined more formally as:

$$(\mathrm{ALLNEG})\ \frac{\mathcal{S}_{\mathrm{p}}(n, t_1) \qquad \mathcal{S}_{\mathrm{p}}(n+1, t_2)}{\mathcal{V}^{-}_{\mathrm{p}}(t)}\ t_1 < t < t_2$$

An additional inference rule for the $\mathrm{LAST_p}$ predicate may now be given as:

$$\text{(LAST*)}\ \frac{\mathcal{S}_\mathrm{p}(n,v) \qquad \mathcal{S}_\mathrm{p}(n+1,w) \qquad \mathcal{V}^+_\mathrm{p}(v)}{\mathrm{LAST_p}(u,v)}\ v \leq u < w$$

Applications may use the $\mathrm{LAST}*$ rule to reduce the cost of validating complex pieces of context through the use of search predicates. If applications assume that the ALLNEG rule holds for the system then the cost of checking the LAST_p predicate becomes constant regardless of the duration of the time period covered.

Unfortunately, applications cannot simply assume that ALLNEG holds because doing so makes an assumption about the validity of the system—the very thing it is supposed to be checking. Thus applications must also validate that the search predicate is producing the correct output and therefore satisfying ALLNEG:

$$\forall t.\ \mathcal{S}_\mathrm{p}(n,t_1) \wedge \mathcal{S}_\mathrm{p}(n+1,t_2) \wedge t_1 < t < t_2 \Rightarrow \mathcal{V}^-_\mathrm{p}(t) \tag{5.5}$$

Simplistic verification of this equation requires checking all the sighting events between t_1 and t_2. This means that there has been no significant reduction in the effort required for validation. However, for specific predicates, a small alteration to the system architecture allows a more simple verification process for this equation.

In the case of the ID selection predicate, it is sufficient to alter the AM node such that it produces a count of the number of times each identifier has been polled. This is achieved by emitting events of the form $(t, \{i, c\})$ for timeslot t indicating the number of times (c) that the identifier (i) has been polled. This specification is incorporated into the logical model of the system by augmenting the existing specification for the trusted nodes (Equations 5.2–5.4) with an additional assumption:

$$\begin{aligned}\forall t_1 t_2 tc.\quad & \mathcal{E}_\mathrm{AM}(t_1,\{i,c\}) \ \wedge\ \mathcal{E}_\mathrm{AM}(t_2,\{i,c+1\}) \ \wedge\ t_1 < t < t_2 \\ & \Rightarrow \\ & \exists jd.\mathcal{E}_\mathrm{AM}(t,\{j,d\}) \ \wedge\ j \neq i\end{aligned}$$

Given this additional specification of system operation, the search predicate $\mathcal{S}_\mathrm{ID}$ may now be defined:

$$\mathcal{S}_\mathrm{ID}(t,n) \equiv \mathcal{E}_\mathrm{AM}(t,\{i,n\}) \wedge i = \mathsf{acr31}$$

Application of this optimisation strategy provides a clear benefit to the application by reducing the costs of validating system operation. This is at the expense of additional complication of the system behaviour. The optimisations available depend on the nature of the underlying system. In Cantag, one cannot define as search predicate analagous to the one above because it any tag could be sighted at any time. The Active Bat system is an instance of a polled system and so makes a decision about when particular Bats will be tracked. Note, that validation cannot

$$
\begin{array}{ll}
{\scriptstyle (\mathrm{LAST}_1)} & \big(\mathcal{V}^+_{\mathrm{p}}(t) \Rightarrow \mathrm{LAST}_{\mathrm{p}}(t,t)\big) \\
& \wedge \\
{\scriptstyle (\mathrm{LAST}_2)} & \big(\mathrm{LAST}_{\mathrm{p}}(t-1,t_p) \wedge \mathcal{V}^-_{\mathrm{p}}(t) \Rightarrow \mathrm{LAST}_{\mathrm{p}}(t,t_p)\big) \\
& \wedge \\
{\scriptstyle (\mathrm{ALLNEG})} & \big(\mathcal{S}_{\mathrm{p}}(n,t_1) \wedge \mathcal{S}_{\mathrm{p}}(n+1,t_2) \wedge t_1 < t < t_2 \Rightarrow \mathcal{V}^-_{\mathrm{p}}(t)\big) \\
& \Rightarrow \\
{\scriptstyle (\mathrm{LAST}*)} & (\mathcal{S}_{\mathrm{p}}(n,t_p) \wedge \mathcal{S}_{\mathrm{p}}(n+1,t_2) \wedge t_p \leq t < t_2 \wedge \\
& \mathcal{V}^+_{\mathrm{p}}(t_1) \Rightarrow \mathrm{LAST}_{\mathrm{p}}(t,t_p)) \\
& \wedge \\
{\scriptstyle (\mathrm{LAST}*_2)} & (\mathcal{S}_{\mathrm{p}}(n,t) \wedge \mathcal{S}_{\mathrm{p}}(n+1,t_2) \wedge t \leq t < t_2 \wedge \\
& \mathcal{V}^-_{\mathrm{p}}(t_1) \wedge \mathrm{LAST}_{\mathrm{p}}(t_1,t_p) \Rightarrow \mathrm{LAST}_{\mathrm{p}}(t,t_p))
\end{array}
$$

Figure 5.16: Specification of validity proof for $(\mathrm{LAST}*)$ and $(\mathrm{LAST}*_2)$

determine whether this is the correct decision and the most suitable Bat has been polled because the polling decision occurs in the trusted area-manager (AM) component.

Finally, the $\mathrm{LAST}*$ rule does not replace the existing rules in the system because it cannot prove everything which is verifiable through the use of the LAST_1 and LAST_2 rules. For example, if acr31 is polled by the AM but no position is successfully resolved then $\mathcal{V}^+_{\mathrm{ID}}$ will not be true and so the application will be unable to use $\mathrm{LAST}*$ to validate the last known position of the application.

An additional inference rule permits suitable reasoning in this situation:

$$
{\scriptstyle (\mathrm{LAST}*_2)}\ \frac{\mathcal{S}_{\mathrm{p}}(n,t_1) \quad \mathcal{S}_{\mathrm{p}}(n+1,t_2) \qquad \mathcal{V}^-_{\mathrm{p}}(t_1) \qquad \mathrm{LAST}_{\mathrm{p}}(t_1,t_p)}{\mathrm{LAST}_{\mathrm{p}}(t,t_p)}\ t_1 \leq t < t_2
$$

5.7.2 Ensuring system safety

As can been seen from the example above it is possible to continue adding additional inference rules to the system in order to improve performance in particular cases. This must be done with caution because validation frameworks aim to improve the dependability of a system. Any transformation is counter-productive if it introduces new errors.

The formalised notation used to describe the validation and the checks performed by the application can be exploited to prove that new rules and optimisations do not permit false inferences and inconsistencies.

To show the consistency of the system including the new $\mathrm{LAST}*$ inference rules it suffices to show that the new rules logically follow from the existing rules in the system. In this situation the new rules are completely defined within the existing proof system. This means they cannot be used to prove anything unprovable in the original system and so, in particular, cannot introduce unsoundness. This proof of soundness equates to demonstrating the truth of the hypothesis in Figure 5.16.

In the interests of conciseness the proof of this hypothesis is omitted in favour of a specification file for the theorem prover Isabelle/HOL [103]. This is shown in Figure 5.17. The proof requires mathematical induction only on the time interval between t_1 and t_2 due to the assumptions

```
theory Bat
imports Main
begin

theorem laststar:
  ⟦ ∀t. VPP t ⟶ LASTP(t,t) ;
    ∀t tp. LASTP(t,tp) ∧ VPM (Suc t) ⟶ LASTP(Suc t,tp) ;
    ∀t te n. SP(n,0) ∧ SP(Suc n,te) ∧ 0 < t ∧ t < te ⟶ VPM t ⟧
  ⟹
   (∀t te n. SP(n,0) ∧ SP(Suc n,te) ∧ 0 ≤ t ∧ t < te ∧ VPP 0 ⟶ LASTP(t,0))
  ∧
   (∀t n te tp. SP(n,0) ∧ SP(Suc n,te) ∧ 0 ≤ t ∧ t < te ∧ VPM 0 ∧ LASTP(0,tp)
     ⟶ LASTP(t,tp))
  apply(subgoal-tac
   ∀ n te.
     (∀t. SP(n,0) ∧ SP(Suc n,te) ∧ 0 ≤ t ∧ t < te ∧ VPP 0 ⟶ LASTP(t,0))
   ∧
     (∀t tp. SP(n,0) ∧ SP(Suc n,te) ∧ 0 ≤ t ∧ t < te ∧ VPM 0 ∧ LASTP(0,tp)
       ⟶ LASTP(t,tp))
  )
  apply(blast)
  apply(rule allI, rule allI)
  apply(subgoal-tac
   ⟦ ∀t. SP(n,0) ∧ SP(Suc n,te) ∧ 0 < t ∧ t < te ⟶ VPM t ⟧
   ⟹
     (∀t. SP(n,0) ∧ SP(Suc n,te) ∧ 0 ≤ t ∧ t < te ∧ VPP 0 ⟶ LASTP(t,0))
   ∧
     (∀t tp. SP(n,0) ∧ SP(Suc n,te) ∧ 0 ≤ t ∧ t < te ∧ VPM 0 ∧ LASTP(0,tp)
       ⟶ LASTP(t,tp))
  )
  apply(blast)
  apply(rule conjI, rule allI)
  apply(induct-tac t)
  apply(simp-all)
  apply(rule allI, rule allI)
  apply(induct-tac t)
  apply(simp-all)
  done
end
```

Figure 5.17: Proof of soundness for the LAST∗ predicates (Isabelle/HOL)

```
theory SIDImpl
imports Main
begin

theorem sidimp:
  ⟦ ∀ t. ∃ j d. EAM(t,j,d) ; ∀ t. VAM(t) ;
    ∀ t t1 t2 j d c i. EAM(t1,i,c) ∧ EAM(t2,i,Suc c) ∧ t1 < t ∧ t < t2
     ⟶ EAM(t,j,d) ∧ j ≠ i ;
    ∀ t i d. VAM(t) ∧ EAM(t,i,d) ∧ i ≠ acr31 ⟶ VIDM(t) ⟧
  ⟹
    ∀ t i n t1 t2. EAM(t1,i,n) ∧ EAM(t2,i,Suc n) ∧ i = acr31 ∧
     t1 < t ∧ t < t2 ⟶ VIDM(t)

  apply(blast)
  done
end
```

Figure 5.18: Proof of validity for the implementation of $\mathcal{S}_{\mathrm{ID}}$ (Isabelle/HOL)

$\mathcal{S}_{\mathrm{p}}(n, t_1)$ and $\mathcal{S}_{\mathrm{p}}(n+1, t_2)$ in the conclusion. Without loss of generality the theorem is restated with a time offset such that $\mathcal{S}_{\mathrm{p}}(n, 0)$ and $\mathcal{S}_{\mathrm{p}}(n+1, t_2 - t_1)$. This permits the application of the induction tactic in Isabelle by recasting the induction to range between 0 and $t_2 - t_1$.

In a similar vein it is also possible to prove that the implementation of $\mathcal{S}_{\mathrm{ID}}$ is correct. This is shown by showing that the additional AM assumption due to $\mathcal{S}_{\mathrm{ID}}$ combined with the trusted component assumptions of the AM (Equation 5.3) and the negative validation rule ID_1^- implies the correctness of the ALLNEG predicate for the region of time of interest (Equation 5.5). The proof of this is shown in Figure 5.18 in which the definition of $\mathcal{S}_{\mathrm{ID}}$ has been substituted for its expanded form: $\mathcal{E}_{\mathrm{AM}}(t, \{i, n\}) \wedge i = acr31$. It is worth noting that this final proof is a nicety that arises from the logical embedding of the Active Bat system—it is unlikely that this proof will be possible for more intricate search predicates.

The rules which an application may use for validation may be viewed as axioms of a logical reasoning system. Applications must trust that the system defined by these axioms is consistent and expresses the intent of validation. The ability to prove that new rules do not permit validation beyond that possible with the original rule-set is a powerful tool because the application does not have to implicitly trust these new (and probably more complex rules). Instead, these new rules may be validated against the original axioms of the system.

Use of an automated theorem prover for these results permits users of the optimised predicates to check for themselves that the optimisations are valid and do not introduce inconsistencies. This is analagous to the vision of Proof-carrying Code [101] in which programs are distributed with an outline for a proof of correctness which may be verified in a theorem prover prior to execution.

5.8 Usage Modes

Its is often the case that the cost of validation will be too high for an application to validate all outputs of a distributed system for errors. However, there are a number of tradeoffs which applications can exploit to approach this ideal.

Applications may elect to statistically check some proportion of the system results based on application-specific criteria. High security or high reliability applications (such as fire warning systems or door locking systems) might use frequent checks to ensure a high probability of failure detection. Applications making soft operational guarantees (such as notification that fresh coffee is available) might accept a lower probability of failure detection.

Stratified sampling of events allows applications to maximise the effect of the validation checks. For example, when validating $\mathrm{LAST}_{\mathrm{p}}(t, t_p)$ an application will provide better operational guarantees if it checks $\mathcal{V}_{\mathrm{p}}^{+}(t_p)$ for the last location sighting that was made (from rule LAST_1) rather than checking $\mathcal{V}_{\mathrm{p}}^{-}(u)$ for some intervening timeslot (u) between t_p and t (from rule LAST_2). This corresponds, in general, to preferring positive validation checks over negative checks.

Administrators might choose to deploy fixed high-power nodes into the network which perform continuous validation of system operation (or some subset thereof). These nodes are not subject to the same constraints in bandwidth and power that a mobile application must abide by.

Network test nodes might check all events from the system or perform statistical testing based on the needs of the currently deployed applications.

If a user of an application is provided with an interface with which to indicate failure then validation can be used to produce a failure report for post-hoc analysis. Personal experience in using and managing a Sentient Computing environment indicates that the users of applications often identify system inconsistencies and errors when they occur.

User-triggered testing may also be configured to evaluate the rejection heuristics in the system. If a user asserts that a failure is due to a false negative then system traces may be re-executed with different rejection heuristics to help identify the fault which caused it. Development of suitable interfaces for this is a topic requiring further investigation.

5.9 Implementation Considerations

Minimising the size of each trusted component gives a number of advantages for validation. Firstly, the amount of code which is uncheckable through validation is reduced. Secondly, if the system produces an invalid output (perhaps this fact is asserted by a user of the system) and validation of the data succeeded then the system administrators can be confident that the source of the error is in the trusted components of the systems or the inputs thereof.

It has also been demonstrated through the optimisation of the LAST predicate that judicious addition of new functionality to the trusted components of the system can have large benefits to application performance.

These observations suggest that designers should minimize the size and functionality of the trusted components of the system before judiciously adding in additional features to meet the needs of applications.

The primary challenge when implementing validation for a system is that of specifying the acceptance tests. This problem is common throughout hardware and software verification. An example of this occurred when implementing validation for Cantag's contour follower. The acceptance test checked that all the points returned as a contour actually lay on the edge of a shape and that no unreported edges remained in the original image. It was only subsequently

discovered that the contour follower was (in certain cases) following each contour in the image twice—a failure mode which still passed the test criteria but which had a marked performance impact.

It is also the case that for most tests it is possible (with fore-knowledge of the test) to engineer a result which will pass the test but not represent the original data. For example, the contour follower test above would also accept a set of one pixel contours which enumerated all edge transitions in the image. This makes the use of validation to protect against malicious alteration of data problematic.

Existing techniques for reliability such as multiple-redundant software components and software verification can be used in tandem with validation. Validation provides the additional advantage to these systems that when the system fails it is easy to locate the responsible component. Additionally, reverse checking of the result from the data is a realistic means to achieve an independent check of an algorithm's result because it forces an entirely different implementation of the computed function.

5.9.1 Validating historical events

Validation of temporal context such as the *entry* event requires validating events which have occurred some arbitrary time in the future. Up to this point the tuple-space for events has been assumed capable of storing all events for all time. One simple method for implementing this is to propagate all events (including intermediate events) through the network to all applications. Discarding unneeded events may then be done in an application-specific manner. However, there are serious efficiency drawbacks to this approach particularly for applications running on low-power, mobile devices.

A more practical implementation is to include a number of network repositories of events which applications can query for events of interest. These repositories will still need to discard events due to limited storage capacity. Events might be discarded on an oldest-first basis or more intelligently with reference to the contextual specifications of the running applications or in keeping with statistical validation efforts.

The investigated deployment of the Active Bat system covers an area of approximately $500\mathrm{m}^2$ and has a total of 409 ceiling receivers spread over 23 Matrix Managers. Each Bat identifier is 48 bits [2] and the timeslot number occupies 80 bits (extended from its original size of 35 bits [145, p49]). The maximum amount of data to be stored for validation per timeslot is therefore:

Timeslot number	80
Polled bat identifier	48
1 distance value per receiver	409×32
x,y,z value per localiser	$23 \times (32 + 32 + 32)$
ID,x,y,z value	$96 + (32 + 32 + 32)$
Total	15616 bits

Each timeslot in the system is approximately $2\mathrm{ms}$ in duration. This equates to approximately 9×10^6 bits per second. This is roughly 30 hours of data for $100\ \mathrm{Gb}$ of storage and so it is

feasible to imagine that discarding data on an oldest-first basis would leave ample historical information for most applications.

The storage costs for Cantag are substantially higher than this. One instance of the system uses a 640×480 pixel resolution camera producing 8-bit grey-scale images at a rate of 50 frames-per-second. The storage of this alone is approximately 100×10^6 bits per second. This is of the order of one hour per $100\ \mathrm{Gb}$ of storage. It is conceivable for applications to wish to reason over longer periods than this and so data acquired at this rate needs to be selectively discarded.

There is potential here for application of data compression techniques to reduce the storage burden. Standard lossless data compression techniques are likely to be highly effective on the highly redundant data from the Active Bat system and the intermediate data from Cantag. Inter-frame compression techniques such as MPEG encoding might also be applicable to data because it is likely that many sightings will be similar to the previous timeslot. For example, in Cantag a moving tag will generate a translated sequence of contours. The affect of lossy coding algorithms (such as MPEG) on the validation process would need to be estimated prior to using this technique.

5.10 Summary

Validation is a useful tool for improving the reliability of operation of an entire context-aware system. In many cases cheap checks are available for the results computed by functional units in the system. Validation also provides a general mechanism for integrating specific checks on a system's observable metrics directly into the relevant part of the system. Validation architectures for Cantag and the Active Bat system have been presented. Together these systems demonstrate a number of important features of location systems which must be accommodated by validation frameworks.

Various optimisations to validation are possible. These optimisations are of general use because they provide means to reduce the number of checks required regardless of their implementation. The strategies for improving performance take a number of different forms. The addition of heuristics for discarding invalid data early in the processing pipeline can be used to direct validation costs away from more expensive checking functions. Basing these heuristics on the theoretical understanding of system behaviour (algorithmic dependability) allows designers to avoid the possibility of introducing false negatives.

It is possible to transform the trusted components of the system in order to permit more efficient validation of particular pieces of information. The formalism developed for expressing validation has been applied in this context and used to show the soundness of an example transformation which reduces the validation cost of the LAST predicate. This allows designers to increase the complexity of the validation rules to efficiently support particular applications whilst guaranteeing the safety of these transformations.

Faults which were internal to the system and difficult to track down are explicitly externalised by the validation approach. This allows users and administrators to identify suspect components if the system produces invalid outputs. Validation also formalises the extent to which an application may ignore partial failure of the underlying system increasing system availability as well as dependability.

Chapter 6

Conclusion

The quantity, complexity, and heterogeneity of devices we expect to manage and operate continues to increase. Reducing the cognitive load imposed by the use of these devices is a growing issue. Sentient Computing systems have the potential to fulfil this goal by coexisting with people in the real, physical environment. However, the unconstrained, continually changing environment (and people within it) creates particular problems for system designers causing failures during the runtime operation of the system. These failures mean that must invest time discerning the status of the system and repairing its faults rather than intuitively interacting with the system. Dependability aims to solve this issue by detecting faults and allowing applications to adapt to them.

6.1 Investigation Platform

This work began with the development of Cantag, a marker-based vision (MBV) location system for Sentient Computing. A significant achievement in the implementation of Cantag was the provision of transparent operation. Transparent operation allows entities outside the system to view internal processing state and results. This is turn aids in the location of faults and allows monitoring of the performance of particular processing algorithms. Cantag is freely available, open-source software, and embodies a robust, reliable platform which is available for future research into Sentient Computing.

The use of the OpenGL test harness integrated into Cantag highlighted some necessary refinements to some of the approaches used in MBV systems. The concept of rotational invariance was provided as an abstraction which allows designers of new fiducial marker tags to apply existing mathematical coding techniques to their tags. The TransformEllipseFull algorithm provides significant improvement to the decoding techniques for circular tags. This algorithm is an extension of an existing “Pose from Circle” algorithm. Significant ambiguities in the original algorithm were identified and rectified.

6.2 Performance Metrics

Algorithmic dependability has been introduced as the study of the theoretical behaviour of a system. Two classes of metric have been identified for describing the performance of a location system:

- **Predictive metrics** allow evaluation of the suitability of the current system deployment for the set of desired applications;
- **Observable metrics** permit runtime evaluation of system performance by estimating performance from the observed data without relying on unknown ground-truth values.

A high-level, information theoretic analysis provided the predictive *sample distance* metric which describes the current tag's amenability to decoding. This metric applies to all marker-based vision systems which operate on 1-bit black and white image data. Optimised layouts for circular tag designs were deduced from this insight.

The more specialised *sample strength* metric incorporates errors in the estimated positions of the datacells on the tag. An observable equivalent to this metric was hypothesised and validated both in simulation and with real-world data. The absolute location accuracy of Cantag was also investigated both in simulation and in with real world data. A dependable tag design and processing pipeline was identified whose real world performance most closely agrees with its predicted behaviour.

6.3 Validation

Observable metrics should be provided to applications at run-time for evaluation of the system's current behaviour. However, insertion of these tests into a running system requires in-depth knowledge of the system. Furthermore, due to time, space, and cost restrictions other sources of failure such as algorithmic errors and implementation problems are likely to still occur in real systems.

Validation provides a means for evaluating these system-specific metrics as well as executing low-cost checks on the intermediate results produced by the system.

The reasoning process required for data validation is conveniently performed by logic programming languages. An example implementation has been described using Prolog extended with suitable external predicates for performing the checks. The costs of validation are strongly dependent upon the number of entities requiring validation checks as they flow through the system. Particular optimisations to the reasoning process have been demonstrated which ameliorate these costs. The first approach is to augment the processing pipeline with heuristics identified from the algorithmic analysis of the system. Additionally, logical transformations of the validation process can have a significant impact by exploiting particular features of the system in question. The application of automated theorem provers for proving the correctness of these transformations was demonstrated in order to ensure that these more complex transformations do not detract from system correctness.

6.4 Future Work

Investigating the dependability of a system is necessarily a process specific to the system itself. At this level, there remain further avenues of investigation with Cantag itself. In particular, more investigation is possible into detecting lighting variations and understanding its effects: these changes seem to cause a dilation or erosion of the target bullseye of the tag and so should be detectable because they alter the relative radii of the edges. Cantag can also provide the platform for further comparison of the performance of image processing algorithms or for the evaluation of newly proposed procedures.

The initial focus for dependability has been into stateless systems without feedback. The next step is to examine the limited feedback which occurs in some systems. In the Active Bat system, for example, there is a feedback path from applications which is used to request an increased polling rate for particular Bats. It will also be interesting to extend the algorithmic dependability analysis for a system in order to derive suitable a priori probabilities for use in stateful filters. Predictive and observable metrics for the outputs of these filters are likely to be significantly more complex due to the importance of historical data.

Currently it is hard to reconcile the level of transparent operation and system knowledge required for dependability with current practices of system abstraction and location system independent middleware. Validation may provide a means to achieve this through its formalism of the reasoning process. The application of validation to other systems and applications may provide additional insights into how to practically achieve this.

A further step on the road to a dependable system is to examine application adaption. Application-specific mechanisms for responding to problems detected through validation must be provided. The benefits of displaying uncertainty directly to users have begun to be investigated by researchers [8]. The benefit of exposing this information versus the increased cognitive load it imposes requires additional study.

6.5 Summary

Currently, Sentient Computing is failure prone due to the difficulties of interacting with users in the physical environment. Dependability is best provided from the lowest level of the system and so should be considered by designers at the early stages of system conception. It is certainly the case that the difficulty of building and implementing real, reliable systems should not be overlooked or underestimated. However, this must not be allowed to mask other fundamental issues such as algorithmic instability or sensing difficulties. Transparency is an important concept when tackling this issue. The application of the approaches and concepts developed in this work have an important role to play in Sentient Computing by helping users to intuitively understand the behaviour of devices and how to interact with the system when failures occur.

References

[1] M.D. Addlesee, A.H. Jones, F. Livesey, and F.S. Samaria. The ORL Active Floor. *IEEE Personal Communications*, 4(5):35–41, October 1997. (Ref: p. 21, 25.)

[2] Mike Addlesee, Rupert Curwen, Steve Hodges, Joe Newman, Pete Steggles, Andy Ward, and Andy Hopper. Implementing a sentient computing system. *Computer*, 34(8):50–56, 2001. (Ref: p. 16, 22, 27, 133.)

[3] Noha Adly, Pete Steggles, and Andy Harter. SPIRIT: a resource database for mobile users. In *Proceedings of ACM CHI'97 Workshop on Ubiquitous Computing*, Atlanta, Georgia, 1997. (Ref: p. 109.)

[4] S.J. Ahn and H.-J. Warnecke. Systematic geometric image measurement errors of circular object targets: Mathematical formulation and correction. *Photogrammetric Record*, 16(93):485–502, April 1999. (Ref: p. 61.)

[5] Andrei Alexandrescu. *Modern C++ Design: Generic Programming and Design Patterns Applied*. Addison-Wesley Professional, 2001. (Ref: p. 53.)

[6] T. Anderson. *Resilient Computer Systems*, chapter 1, pages 1–10. Collins, 1985. (Ref: p. 40.)

[7] Michael Angermann, Jens Kammann, Patrick Robertson, Alexander Steingaß, and Thomas Strang. Software representation for heterogenous location data sources using probability density functions. In *International Symposium on Location Based Services for Cellular Users*, 2001. (Ref: p. 37, 79.)

[8] Stavros Antifakos, Adrian Schwaninger, and Bernt Schiele. Evaluating the effects of displaying uncertainty in context-aware applications. In *UbiComp2004: Ubiquitous Computing*, pages 54–69, 2004. (Ref: p. 137.)

[9] Hisashi Aoki, Bernt Schiele, and Alex Pentland. Realtime personal positioning system for wearable computers. In *ISWC '99: Proceedings of the 3rd IEEE International Symposium on Wearable Computers*, pages 37–43, 1999. (Ref: p. 31.)

[10] ASTM. G173-03e1 standard tables for reference solar spectral irradiances: Direct normal and hemispherical on 37^o tilted surface. (Ref: p. 30.)

[11] P. Bahl, A. Balachandran, and V. Padmanabhan. Enhancements to the RADAR user location and tracking system. Technical Report MSR-TR-2000-12, Microsoft Research, 2000. (Ref: p. 33.)

[12] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *INFOCOM 2*, pages 775–774, 2000. (Ref: p. 33.)

[13] H. E. Bass, L. C. Sutherland, A. J. Zuckerwar, D. T. Blackstock, and D. M. Hester. Atmospheric absorption of sound: Further developments. *The Journal of the Acoustical Society of America*, 97(1):680–683, 1995. (Ref: p. 27.)

[14] Sassan Bassiri and George A. Hajj. Higher-order ionospheric effects on the GPS observables and means of modeling them. *NASA STI/Recon Technical Report A*, 95, 1993. (Ref: p. 37.)

[15] Steve Benford, Rob Anastasi, Martin Flintham, Adam Drozd, Andy Crabtree, Chris Greenhalgh, Nick Tanavanitj, Matt Adams, and Ju Row-Farr. Coping with uncertainty in a location-based game. *Pervasive Computing*, 2(3):34–41, July 2003. (Ref: p. 19.)

[16] Steve Benford, Will Seager, Martin Flintham, Rob Anastasi, Duncan Rowland, Jan Humble, Danaë Stanton, John Bowers, Nick Tandavanitj, Matt Adams, Ju Row Farr, Amanda Oldroyd, and Jon Sutton. The error of our ways: The experience of self-reported position in a location-based game. In *UbiComp2004: Ubiquitous Computing*, pages 70–87, 2004. (Ref: p. 32.)

[17] C. Leonard Bennett and Gerald F. Ross. Time-domain electromagnetics and its applications. *Proceedings of the IEEE*, 66(3):299–318, 1978. (Ref: p. 34.)

[18] Alastair R. Beresford. *Location privacy in ubiquitous computing*. PhD thesis, University of Cambridge, 2005. (Ref: p. 16.)

[19] Xuehai Bian, Gregory D. Abowd, and James M. Rehg. Using sound source localization in a home environment. In *Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive 2005)*, pages 19–36, May 2005. (Ref: p. 24.)

[20] Mark Billinghurst, Hirkazu Kato, and Ivan Poupyrev. The MagicBook—moving seamlessly between reality and virtuality. *IEEE Computer Graphics and Applications*, 21(3):6–8, 2001. (Ref: p. 47.)

[21] Mark Billinghurst and Hirokazu Kato. Collaborative mixed reality. In *Proceedings of the First International Symposium on Mixed Reality*, pages 261–284, 1999. (Ref: p. 48, 49.)

[22] Dennis A. Bohn. Environmental effects on the speed of sound. *Journal of the Audio Engineering Society*, 34(4):223–231, April 1988. (Ref: p. 27.)

[23] Gaetano Borriello, Alan Liu, Tony Offer, Christopher Palistrant, and Richard Sharp. WALRUS: wireless acoustic location with room-level resolution using ultrasound. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services (MobiSys '05)*, pages 191–203, 2005. (Ref: p. 28.)

[24] Frederick P. Brooks Jr. *The Mythical Man-Month*. Addison Wesley Longman, Inc, 1999. (Ref: p. 15.)

[25] Aaron B. Brown and David A. Patterson. Rewind, repair, replay: Three R's to dependability. In *10th ACM SIGOPS European Workshop*, September 2002. (Ref: p. 43.)

[26] D.C. Brown. Decentering distortion of lenses. *Photogrammetric Engineering and Remote Sensing*, 32(3):444–462, 1966. (Ref: p. 31, 58.)

[27] James Brusey, Christian Floerkemeier, Mark Harrison, and Martyn Fletcher. Reasoning about uncertainty in location identification with RFID. In *IJCAI-03 Workshop on Reasoning with Uncertainty in Robotics*, Acapulco, Mexico, August 2003. (Ref: p. 33.)

[28] Sung H. Byun, George A. Hajj, and Lawrence E. Young. Development and application of GPS signal multipath simulator. *Radio Science*, 37(6), 2002. (Ref: p. 37.)

[29] George Candia and Armando Fox. Designing for high availability and measurability. In *Proceedings of the 1st Workshop on Evaluating and Architecting System Dependability*, 2001. (Ref: p. 42.)

[30] Mike Y. Chen, Emre Kiciman, Eugene Fratkin, Armando Fox, and Eric Brewer. Pinpoint: Problem determination in large, dynamic internet services. In *DSN '02: Proceedings of the 2002 International Conference on Dependable Systems and Networks*, pages 595–604, 2002. (Ref: p. 43.)

[31] Youngkwan Cho, Jongweon Lee, and Ulrich Neumann. A multi-ring color fiducial system and an intensity-invariant detection method for scalable fiducial-tracking augmented reality. In *Proceedings of the First International Workshop in Augmented Reality*, pages 147–165, 1998. (Ref: p. 48.)

[32] Youngkwan Cho and Ulrich Neumann. Multiring fiducial systems for scalable fiducial-tracking augmented reality. *PRESENCE: Teleoperators and Virtual Environments*, 10(6):599–612, December 2001. (Ref: p. 63.)

[33] Carmine Ciavarella and Fabio Paternò. The design of a handheld, location-aware guide for indoor environments. *Personal and Ubiquitous Computing*, 8(2):82–91, 2004. (Ref: p. 19, 22.)

[34] John Daugman and Cathryn Downing. Epigenetic randomness, complexity and singularity of human iris patterns. *Proceedings. Biological sciences / The Royal Society*, 268(1477):1737–1740, 2001. (Ref: p. 24.)

[35] Diego López de Ipinã. *Visual Sensing and Middleware Support for Sentient Computing*. PhD thesis, University of Cambridge, 2002. (Ref: p. 62, 71, 84.)

[36] Diego López de Ipinã. A low-cost vision-based location system for ubiquitous computing. *Personal and Ubiquitous Computing Journal*, 6(3):206–219, May 2002. (Ref: p. 48, 76.)

[37] Diego López de Ipiña, Paulo R. S. Mendonça, and Andy Hopper. TRIP: a low-cost vision-based location system for ubiquitous computing. *Personal and Ubiquitous Computing*, 6(3):206–219, May 2002. (Ref: p. 24, 32.)

[38] George Dedes and Andrew G. Dempster. Indoor GPS positioning—challenges and opportunities. In *IEEE 62nd Vehicular Technology Conference*, volume 1, pages 412–415, 2005. (Ref: p. 34.)

[39] Frank Dellaert, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Using the Condensation algorithm for robust, vision-based mobile robot localization. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)*, 21(2):208–216, June 1999. (Ref: p. 31.)

[40] Anind K. Dey and Gregory D.Abowd. Towards a better understanding of context and context-awareness. In *Proceedings of the CHI 2000 Workshop on "The What, Who, Where, When, Why and How of Context-Awareness"*, 2000. (Ref: p. 16, 19.)

[41] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973. (Ref: p. 59.)

[42] Charles E. Ebeling. *An introduction to reliability and maintainability engineering*. McGraw-Hill, 1997. (Ref: p. 39, 42.)

[43] David Patterson et al. Recovery oriented computing (ROC): Motivation, definition, techniques, and case studies. Technical Report UCB//CSD-02-1175, UC Berkeley, March 2002. (Ref: p. 42.)

[44] Howard Eves. *A Survey of Geometry*. Allyn and Bacon Incorporated, 1972. (Ref: p. 50.)

[45] Gerald Farin and Dianne Hansford. *The Geometry Toolbox For Graphics and Modelling*. A K Peters, 1998. (Ref: p. 59.)

[46] John Fawcett. *Sentient Computing: A Universal Framework for Spatial Data*. PhD thesis, University of Cambridge, 2002. (Ref: p. 22, 27.)

[47] Silke Feldmann, Kyandoghere Kyamakya, Ana Zapater, and Zighuo Lue. An indoor Bluetooth-based positioning system: Concept, implementation and experimental evaluation. In *International Conference on Wireless Networks*, pages 109–113, 2003. (Ref: p. 34.)

[48] F. Figueroa and A. Mahajan. A robust navigation system for autonomous vehicles using ultrasonics. *Control Engineering Practice*, 2(1):49–59, 1994. (Ref: p. 28.)

[49] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981. (Ref: p. 28.)

[50] Kenneth P. Fishkin, Bing Jiang, Matthai Philipose, and Sumit Roy. I sense a disturbance in the force: Unobstrusive detection of interactions with RFID-tagged objects. In *UbiComp 2004: Ubiquitous Computing*, pages 268–282, 2004. (Ref: p. 33.)

[51] Robert Fontana. Experimental results from an ultra wideband precision geolocation system. In *Ultra-Wideband, Short-Pulse Electromagnetics IV*. Kluwer Academic/Plenum Publishers, 2000. (Ref: p. 34.)

[52] Office for National Statistics. *Family Spending—A report on the Expenditure and Food Survey*. Office for National Statistics, 2005. (Ref: p. 15.)

[53] David Forsyth, Joseph L. Mundy, Andrew Zisserman, Chris Coelho, Aaron Heller, and Charles Rothwell. Invariant descriptors for 3-D object recognition and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):971–991, October 1991. (Ref: p. 62, 71.)

[54] Dieter Fox, Jeffrey Hightower, Lin Liao, Dirk Schulz, and Gaetano Borriello. Bayesian filtering for location estimation. *IEEE Pervasive*, 2(3):24–33, July 2003. (Ref: p. 24.)

[55] Eric Foxlin. Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer Graphics and Applications*, 25(6):38–46, 2005. (Ref: p. 26.)

[56] Leonid Gavrilov and Natalia Gavrilova. Why we fall apart: engineering's reliability theory explains human aging. *IEEE Spectrum*, 41(9):31–35, September 2004. (Ref: p. 39.)

[57] David Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 7(1):80–112, January 1985. (Ref: p. 110.)

[58] Ivan Getting. The Global Positioning System. *IEEE Spectrum*, 30(12):36–38,43–47, December 1993. (Ref: p. 21, 34.)

[59] Salvatore Gravano. *Introduction to Error Control Codes*. Oxford University Press, 2001. (Ref: p. 63.)

[60] Radim Halíř and Jan Flusser. Numerically stable direct least squares fitting of ellipses. In *The Sixth International Conference in Central Europe on Computer Graphics and Visualization*, 1998. (Ref: p. 59, 108.)

[61] R.K. Harle and A. Hopper. Building world models by ray-tracing within ceiling-mounted positioning systems. In *Proceedings of UbiComp 2003: Ubiquitous Computing: 5th International Conference*, October 2003. (Ref: p. 22.)

[62] A. Harter and A. Hopper. A distributed location system for the active office. *IEEE Network*, 8(1):62–70, January 1994. (Ref: p. 32.)

[63] Andy Harter and Frazer Bennett. Low bandwidth infra-red networks and protocols for mobile communicating devices. Technical Report 93.5, Olivetti Research Limited, 1993. (Ref: p. 29.)

[64] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003. (Ref: p. 46, 58.)

[65] Michael D. Hazas. *Indoor Location Systems*. PhD thesis, University of Cambridge, 2002. (Ref: p. 28.)

[66] Mike Hazas and Andy Hopper. Broadband ultrasonic location systems for improved indoor positioning. *IEEE Transactions on Mobile Computing*, 5(5):536–547, 2006. (Ref: p. 28.)

[67] Robert Headon and Rupert Curwen. Recognising movements from the ground reaction force. In *Workshop on Perceptual User Interfaces*, November 2001. (Ref: p. 16.)

[68] Glenn E. Healey and Raghava Kondepudy. Radiometric CCD camera calibration and noise estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(3):267–276, March 1994. (Ref: p. 35.)

[69] Herbert Hecht and Myron Hecht. *Fault-Tolerant Computer System Design*, chapter 7, pages 428–476. Prentice-Hall, 1996. (Ref: p. 42.)

[70] Karen Henricksen and Jadwiga Indulska. Modelling and using imperfect context information. In *Second IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 33–37, 2004. (Ref: p. 36.)

[71] Jeffrey Hightower and Gaetano Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, August 2001. (Ref: p. 36.)

[72] Jeffrey Hightower and Gaetano Borriello. Real-time error in location modeling for ubiquitous computing. In *Location Modeling for Ubiquitous Computing—Ubicomp 2001 Workshop*, pages 21–27, September 2001. (Ref: p. 36, 37, 79.)

[73] Martin Hiller. Executable assertions for detecting data errors in embedded control systems. In *International Conference on Dependable Systems and Networks (DSN 2000)*, pages 24–36, New York, USA, June 2000. (Ref: p. 40.)

[74] Martin Hiller, Arshad Jhumka, and Neeraj Suri. On the placement of software mechanisms for detection of data errors. In *Proceedings of the 2002 International Conference on Dependable Systems and Networks*, pages 135–144, 2002. (Ref: p. 41.)

[75] Andy Hopper. The Clifford Paterson lecture 1999: Sentient computing. *Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences*, 358(1773):2349–2358, 2000. (Ref: p. 16.)

[76] K.H. Huang and J.A. Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE Transactions on Computers (Special Issue on Reliable and Fault-Tolerant Computing)*, C-33(6):518–528, June 1984. (Ref: p. 41.)

[77] Don R. Hush and Cliff Wood. Analysis of tree algorithms for RFID arbitration. In *Proceedings of IEEE International Symposium on Information Theory*, pages 107–117, 1998. (Ref: p. 33.)

[78] Barry W. Johnson. *Fault-Tolerant Computer System Design*, chapter 1, pages 1–84. Prentice-Hall, 1996. (Ref: p. 38.)

[79] Rana El Kaliouby and Peter Robinson. Generalization of a vision-based computational model of mind-reading. In *Affective Computing and Intelligent Interaction, First International Conference*, pages 582–589, 2005. (Ref: p. 32.)

[80] Kenichi Kanatani. *Geometric Computation for Machine Vision*, chapter 8, pages 246–249. Clarendon Press, 1993. (Ref: p. 71, 72, 73.)

[81] Tim Kindberg and John Barton. A web-based nomadic computing system. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 35(4):443–456, 2001. (Ref: p. 29.)

[82] Georg S. W. Klein and Tom Drummond. Robust visual tracking for non-instrumented augmented reality. In *IEEE / ACM International Symposium on Mixed and Augmented Reality (ISMAR 2003)*, pages 113–122, 2003. (Ref: p. 31.)

[83] Philip Koopman and Tridib Chakravarty. Cyclic Redundancy Code (CRC) polynomial selection for embedded networks. In *2004 International Conference on Dependable Systems and Networks, DSN'04*, pages 145–157, 2004. (Ref: p. 67.)

[84] John Krumm, Steve Harris, Brian Meyers, Barry Brumitt, Michael Hale, and Steve Shafer. Multi-camera multi-person tracking for EasyLiving. In *IEEE Workshop on Visual Surveillance*, 2000. (Ref: p. 19, 24.)

[85] Jaakko Lähteenmäki. Indoor propagation models. In *COST Action 231: Digital mobile radio towards future generation systems*, EUR 18957, chapter 4, pages 175–190. Cooperation in the field of Scientific and Technical Research (COST), 1999. (Ref: p. 34.)

[86] Mik Lamming and Mike Flynn. Forget-Me-Not: Intimate computing in support of human memory. In *Proceedings of the International Symposium on Next Generation Human Interface Technologies*, February 1994. (Ref: p. 22.)

[87] F. Landstorfer, G. Woelfle, and R. Hoppe. Propagation models for indoor communications. In *Wiley Encylopedia of Telecommunications*, volume 4, pages 2012–2021. Wiley, 2003. (Ref: p. 33, 34.)

[88] Seon-Woo Lee and Kenji Mase. Incremental motion-based location recognition. In *Fifth International Symposium on Wearable Computers (ISWC'01)*, pages 123–121, Zurich, Switzerland, October 2001. (Ref: p. 25.)

[89] Hui Lei, Daby M. Sow, John S. Davis II, Guruduth Banavar, and Maria R. Ebling. The design and applications of a context service. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(4):45–55, 2002. (Ref: p. 36.)

[90] Alfred Leick. *GPS Satellite Surveying*. John Wiley & Sons, Inc., 2004. (Ref: p. 36.)

[91] Ulf Leonhardt. *Supporting Location-Awareness in Open Distributed Systems*. PhD thesis, Imperial College, London, 1998. (Ref: p. 22, 37.)

[92] N. Leveson. *Resilient Computer Systems*, chapter 7, pages 122–143. Collins Professional and Technical Books, London, UK, 1985. (Ref: p. 41.)

[93] Anil Madhavapeddy and Alastair Tse. A study of bluetooth propagation using accurate indoor location mapping. In *Seventh International Conference on Ubiquitous Computing (UbiComp '05)*, pages 105–122, August 2005. (Ref: p. 34.)

[94] John Mallon and Paul F Whelan. Precise radial un-distortion of images. In *Proceedings of the 17th International Conference on Pattern Recognition*, pages 18–21, 2004. (Ref: p. 59.)

[95] Joseph F. McCarty and Eric S. Meidel. ActiveMap: A visualization tool for location awareness to support informal interactions. In *Proceedings of the International Symposium on Handheld and Ubiquitous Computing*, 1999. (Ref: p. 20.)

[96] Keith D. McDonald and Christopher Hegarty. Post-modernisation GPS performance capabilities. In *IAIN World Congress in Association with the U.S. ION 56th Annual Meeting*, June 2000. (Ref: p. 37.)

[97] Scott Meyers. *More Effective C++*. Addison-Wesley, 1996. (Ref: p. 110.)

[98] Masateru Minami, Yasuhiro Fukuju, Kazuki Hirasawa, Shigeaki Yokoyama, Moriyuki Mizumachi, Hiroyuki Morikawa, and Tomonori Aoyamaa. DOLPHIN: A practical approach for implementing a fully distributed indoor ultrasonic positioning system. In *UbiComp 2004: Ubiquitous Computing*, pages 347–365, 2004. (Ref: p. 35.)

[99] Migual A. Muñoz, Marcela Rodríguez, Jesus Favela, Ana I. Matinez-Garcia, and Victor M. González. Context-aware mobile communications in hospitals. *IEEE Compouter*, 36(9):38–46, September 2003. (Ref: p. 19.)

[100] Leonid Naimark and Eric Foxlin. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *IEEE International Symposium on Mixed and Augmented Reality*, pages 27–36, September 2002. (Ref: p. 27, 32, 48.)

[101] George C. Necula. Proof-carrying code. In *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Langauges (POPL '97)*, pages 106–119, January 1997. (Ref: p. 131.)

[102] Lionel M. Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P. Patil. LANDMARC: indoor location sensing using active RFID. *Wireless Networks*, 10(6):701–710, 2004. (Ref: p. 34.)

[103] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL—A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer-Verlag, 2002. (Ref: p. 129.)

[104] Robert J. Orr and Gregory D. Abowd. The Smart Floor: A mechanism for natural user identification and tracking. In *Proceedings of the 2000 Conference on Human Factors in Computing Systems (CHI 2000)*, The Hague, Netherlands, April 2000. (Ref: p. 26.)

[105] Charles B. Owen, Fan Xiao, and Paul Middlin. What is the best fiducial? In *The First IEEE International Augmented Reality Toolkit Workshop*, pages 98–105, September 2002. (Ref: p. 48, 49.)

[106] Ganesh J. Pai and Joanne Bechta Dugan. Automatic synthesis of dynamic fault trees from UML system models. In *13th International Symposium on Software Reliability Engineering*, pages 243–256, Maryland, USA, November 2002. (Ref: p. 41.)

[107] Joseph Paradiso, Craig Abler, Kai-yuh Hsiao, and Matthew Reynolds. The Magic Carpet: Physical sensing for immersive environments. In *Proceedings of the 1997 Conference on Human Factors in Computing Systems (CHI '97)*, pages 277–278, 1997. (Ref: p. 26, 39.)

[108] Gopal Pingali, Claudio Pinhanez, Anthony Levas, Rick Kjeldsen, Mark Podlaseck, Han Chen, and Noi Sukaviriya. Steerable interfaces for pervasive computing spaces. In *First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*, pages 315–323, Texas, March 2003. (Ref: p. 21.)

[109] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The Cricket location-support system. In *Mobile Computing and Networking*, pages 32–43, 2000. (Ref: p. 23.)

[110] Nissanka B. Priyantha, Allen K.L. Miu, Hari Balakrishnan, and Seth Teller. The Cricket Compass for Context-Aware mobile applications. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom '01)*, pages 1–14, 2001. (Ref: p. 28.)

[111] Nissanka Bodhi Priyantha. *The Cricket Indoor Location System*. PhD thesis, Massachusetts Institute of Technology, 2005. (Ref: p. 28.)

[112] Lisa Ran, Sumi Helal, and Steve Moore. Drishti: An integrated indoor/outdoor blind navigation system and service. In *Second IEEE International Conference on Pervasive Computing and Communications*, pages 23–33, 2004. (Ref: p. 19.)

[113] B. Randell. System structure for software fault tolerance. In *Proceedings of the international conference on Reliable software*, pages 437–449, 1975. (Ref: p. 42.)

[114] Jun Rekimoto. Matrix: A realtime object identification and registration method for augmented reality. In *Proceedings of Asia Pacific Computer Human Interaction*, pages 63–68, July 1998. (Ref: p. 24, 32, 48, 62, 63.)

[115] Jun Rekimoto and Yuji Ayatsuka. CyberCode: Designing augmented reality environments with visual tags. In *Proceedings of DARE 2000 on Designing augmented reality environments*, pages 1–10, 2000. (Ref: p. 32, 48, 62.)

[116] Andrew Rice, Christopher Cain, and John Fawcett. Dependable coding for fiducial tags (extended version). In *Ubiquitous Computing Systems, LNCS 3598*, pages 259–274, 2004. (Ref: p. 68, 70.)

[117] Andrew Rice and Robert Harle. Evaluating lateration-based positioning algorithms for fine-grained tracking. In *Joint Workshop on Foundations of Mobile Computing (DIAL-M-POMC)*, pages 54–61. ACM Press, 2005. (Ref: p. 28.)

[118] Tristan Richardson, Frazer Bennett, Glenford Mapp, and Andy Hopper. Teleporting in an X Window system environment. In *IEEE Personal Communications*, volume 1, pages 6–12, 1994. (Ref: p. 21, 22.)

[119] Michael Rohs. Real-world interactiong with camera-phones. In *Proceedings of the 2nd Ubiquitous Computing Symposium*, pages 39–49, 2004. (Ref: p. 47, 48.)

[120] Paul L. Rosin. Assessing error of fit functions for ellipses. *Graphical Models and Image Processing*, 58(5):494–502, September 1996. (Ref: p. 76, 109.)

[121] Thomas D. Rossing. *The Science of Sound*. Addison-Wesley Publishing Company, Inc., second edition, 1990. (Ref: p. 27.)

[122] Wasinee Rungsarityotin and Thad E. Starner. Finding location using omnidirectional video on a wearable computing platform. In *ISWC '00: Proceedings of the 4th IEEE International Symposium on Wearable Computers*, pages 61–68, 2000. (Ref: p. 31.)

[123] Daniel Salber, Anind K. Dey, and Gregory D. Abowd. The Context Toolkit: Aiding the development of context-enabled applications. In *Conference on Human Factors in Computing Systems*, pages 434–441, 1999. (Ref: p. 109.)

[124] David Scott, Richard Sharp, Anil Madhavapeddy, and Eben Upton. Using visual tags to bypass bluetooth device discovery. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(1):41–53, 2005. (Ref: p. 24, 47.)

[125] James Scott and Boris Dragovic. Audio location: Accurate low-cost location sensing. In *Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive 2005)*, pages 1–18, May 2005. (Ref: p. 24.)

[126] James Scott and Mike Hazas. User-friendly surveying techniques for location-aware systems. In *Proceedings of the Fifth International Conference on Ubiquitous Computing*, volume 2864, pages 45–54, October 2003. (Ref: p. 35.)

[127] M. R. Shortis and H. A. Beyer. *Close Range Photogrammetry and Machine Vision*, chapter 5, pages 106–155. Whittles Publishing, 1996. (Ref: p. 55.)

[128] P. Srinivasan, D. Birchfield, G. Qian, and A. Kidané. A pressure sensing floor for interactive media applications. In *Proceedings of ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE)*, 2005. (Ref: p. 26.)

[129] T. Starner, D. Kirsch, and S. Assefa. The locust swarm: An environmentally-powered, networkless location and messaging system. In *Proceedings of the 1st IEEE International Symposium on Wearable Computers (ISWC '97)*, page 169, 1997. (Ref: p. 23.)

[130] Thad Starner, Bernt Schiele, and Alex Pentland. Visual contextual awareness in wearable computing. In *ISWC '98: Proceedings of the 2nd IEEE International Symposium on Wearable Computers*, pages 50–57, 1998. (Ref: p. 31.)

[131] Andrei State, Gentaro Hirota, David T. Chen, William F. Garrett, and Mark A. Livingston. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *Proceedings of SIGGRAPH 96*, pages 429–438. ACM, August 1996. (Ref: p. 48.)

[132] Peter Sturm. Algorithms for plane-based pose estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'00)*, volume 1, pages 1706–1717, 2000. (Ref: p. 62.)

[133] S. Suzuki and K. Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985. (Ref: p. 58.)

[134] Cho-Huak Teh and Roland T. Chin. On the detection of dominant points on digital curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(8):859–872, 1989. (Ref: p. 59.)

[135] David Tennenhouse. Proactive computing. *Communications of the ACM*, 43(5):43–50, 2000. (Ref: p. 16.)

[136] G A Thomas, J Jin, and C Urquhart. A versatile camera position measurement system for virtual reality TV production. In *International Broadcasting Convention*, pages 284–289, 1997. (Ref: p. 32, 47.)

[137] Øivind Due Trier and Anil K. Jain. Goal-directed evaluation of binarization methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1191–1201, December 1995. (Ref: p. 57.)

[138] Stephen Urry. Plantar pressure-measurement sensors. *Measurement Science and Technology*, 10(1):R16–R32, 1999. (Ref: p. 25, 26.)

[139] US Army Corps of Engineers. *NAVSTAR Global Positioning System Surveying*, 2003. (Ref: p. 37.)

[140] Kiran Kumar Vemuri, Joanne Bechta Dugan, and Kevin J. Sullivan. Automatic synthesis of fault trees for computer-based systems. *IEEE Transactions on Reliability*, 48(4):394–402, December 1999. (Ref: p. 41.)

[141] Daniel Wagner, Thomas Pintaric, Florian Ledermann, and Dieter Schmalstieg. Towards massively multi-user augmented reality on handheld devices. In *Third International Conference on Pervasive Computing*, 2005. (Ref: p. 20, 47.)

[142] R. Want, B. N. Schilit, N. I. Adams, R. Gold, K. Petersen, D. Goldberg, J. R. Ellis, and M. Weiser. An overview of the PARCTAB Ubiquitous Computing experiment. *IEEE Personal Communications*, 2(6):28–33, Dec 1995. (Ref: p. 29.)

[143] Roy Want, Kenneth P. Fishkin, Anuj Gujar, and Beverly L. Harrison. Bridging physical and virtual worlds with electronic tags. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 370–377, 1999. (Ref: p. 32.)

[144] Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons. The Active Badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, January 1992. (Ref: p. 19, 20.)

[145] Andrew Robert Martin Ward. *Sensor-driven Computing*. PhD thesis, University of Cambridge, 1998. (Ref: p. 19, 27, 28, 133.)

[146] Andy Ward, Alan Jones, and Andy Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47, October 1997. (Ref: p. 21.)

[147] Lauren Weiner. *Digital Woes: why we should not depend on software*. Addison-Wesley, 1993. (Ref: p. 46, 108.)

[148] Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, January 1991. (Ref: p. 15.)

[149] Greg Welch and Eric Foxlin. Motion tracking: No silver bullet, but a respectable arsenal. *IEEE Computer Graphics and Applications*, 22(6), 2002. (Ref: p. 23.)

[150] Pierre Wellner. Adaptive thresholding for the DigitalDesk. Technical Report EPC-93-110, EuroPARC, 1993. (Ref: p. 57.)

[151] Shigeki Yokoi, Jun-Ichiro Toriwaki, and Teruo Fukumura. An analysis of topological properties of digitized binary pictures using local features. *Computer Vision, Graphics, and Image Processing*, 4:63–73, 1975. (Ref: p. 58.)

[152] Xiang Zhang, Stephan Fronz, and Nassir Navab. Visual marker detection and decoding in AR systems: A comparative study. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, pages 97–107, 2002. (Ref: p. 103.)

[153] Xiaowei Zhong, Peiran Liu, Nicolas D. Georganas, and Pierre Boulanger. Designing a vision-based collaborative augmented reality application for industrial training. *it-Information Technology*, 45(1):7–18, 2003. (Ref: p. 48, 63.)