Chapter 7
Sources, Models and Use of Location: *A Special Sort of Context*

Location, as a form of context and as a specialised concern, has been a central consideration of pervasive computing from the start. Location, as a way of indexing data and through distribution of computation, has an even more venerable history in computer science. Of particular interest here is the nature of data which describes the world and associates the virtual with the physical world, and so there is some overlap with Geographical Information Systems (GIS). In this chapter we shall review some of the key ideas about location in computing and their application in pervasive computing.

7.1 Uses of Location

Location has many uses in pervasive computing, helping applications understand questions including:

- Where am I?
- Where is v?
- Who is near?
- Is w within 100 m of me?
- Where should I go to find the nearest x?
- Where was I when y happened?
- How do I get to z?

In all cases the identifier z etc. may identify a unique object in the world, e.g. "John" or "my laptop", specify some class of objects, e.g. "colour printer", or describe a pattern to match, e.g. "co-location with Fred" or "open space of 100–200 m^2". The case of "where is v" highlights an interesting pair of subdivisions: Firstly, whether the object being located is the thing or person that is really of interest, or some proxy, e.g. a person's mobile phone. Secondly, whether the object being located is participating in the act of being located, unaware, or being uncooperative. A participating object will be a computational device, possibly enhanced by location awareness hardware,

D. Chalmers, *Sensing and Systems in Pervasive Computing*,
Undergraduate Topics in Computer Science,
DOI 10.1007/978-0-85729-841-6_7, © Springer-Verlag London Limited 2011

engaging in a protocol to share information. An uncooperative object, often in a military scenario, is actively avoiding detection by blocking signals, camouflaging its presence or providing mis-information. A willing participant may obfuscate their location to manage privacy, which is different to the uncooperative case. An unaware object is neither actively seeking engagement nor hindering observation. It may not yet be aware of some benefit to being located, which may be advertised to it once located; the location of objects may be treated in aggregate, e.g. locating cars in order to identify unoccupied parking spaces; or the located objects may remain unaware of being tracked, e.g. in monitoring animal migration, or resigned to being tracked through some legal obligation, e.g. emergency phone call location or security surveillance. In this book we shall concentrate on those cases where being located allows or enhances engagement with a service.

Location acts both a form of context and as a special sort of data in mobile applications. Our presentation is equally relevant to both interpretations. Our notation of $\mathcal{C}_{a,o}$ still fits: the aspect is location, the object is the *located object*; often in context awareness the primary object of interest is ourselves, but with location awareness we are often also interested in the location of other objects as well. Returning to the types of context awareness from Chap. 5, we can find uses of location for each of them:

1. Context display is often manifested as showing a map of the current location.
2. Context aware configuration may use a model of nearby resources to configure a system. "Nearby" might be physically near, such as a local printer, or near in system terms, such as a proxy-server with a good (high bandwidth, low latency, probably few hops) network connection.
3. Context triggered actions might cause behaviour on reaching a particular place (home, my office, the train) or when location changes, e.g. request a new map when current location is half-way to the edge of the old map.
4. Contextual augmentation may annotate records of events with their location, and the co-location of important others.
5. Contextual mediation may modify data depending on location, or distance from destination.
6. Contextual adaptation of the environment may cause heating and lighting preferences to follow a user, locating the responsible controllers for the current location as it changes.
7. Context aware presentation may modify the display of private information depending on the location of the user, and the presence of others.

By considering possible questions and possible uses of context we have already, informally, raised the possibility of relations (within 100 m, nearest), fuzzy relations (near), route finding, interpreting the local features of a location, dynamic location information (who is where, as well as static roads and buildings), semantic locations (home) and coordinates relative to a reference (show on map). Rather than give a series of solutions, in this chapter we shall set out some well-understood tools that are appropriate to devising solutions to these, and other location-based, problems.

7.2 Coordinate Models

Coordinate models are built into some systems, but an outline of the underlying principles seems appropriate here. Many of the ideas in this section are developed in more detail in standard geography / GIS texts and technical documents such as (the rather UK centric) Ordnance Survey model [20] and product manuals.

7.2.1 GIS

Geographic Information Systems (GIS) have been a subject of research and commercial development for many years. The data sets provided are a rich source of information about the world for pervasive computing to tap. GIS systems typically use *vector* map formats, representing individual features with detailed position and semantic information—rather than the *raster* maps we are more familiar with, where the set of features and rendering are pre-determined. Representing individual features and classes of feature allows customised maps to be produced, combining data sets. This rich description can support many location aware applications beyond the map, by making use of rich models of place and sophisticated data descriptions. The description of features can be sub-categorised:

- *Fields* describe the properties of regions in space, mapping from place to property, even where the boundaries are indistinct. This model lends itself to many typical GIS situations, including aerial photographic data sources and continuous systems of geography and environment.
- *Objects* are an alternative to fields, whose concept will be familiar to computer scientists. In GIS an object describes a physical entity, with defined boundaries, which can be represented on a map. Object models lend themselves to situations where legally defined entities, e.g. land ownership are being described, and where mobile assets, e.g. trucks, parcels, are moving against a largely static backdrop.
- Objects may further be defined as *points*, with a single spatial location; *lines* or *curves*, with start and end points and a path between them; or *polygons*, with a boundary defined by a series of points. This basic model is found in many spatial data definitions, although often with extensions, such as: *Collections*, defined as multiples of one of the other types; relationships, e.g. road lines connected by junction points; and additional attributes, including directionality for a line.

OpenGIS[1] is a popular current GIS description standard which embodies many of these characteristics.

Google Maps[2] provides an easily accessed view of combined raster (satellite) and vector drawings (e.g. roads, although may be presented to the browser as pre-drawn

[1] http://www.opengeospatial.org/.

[2] http://maps.google.com.

rasters) with other data sets (e.g. images, traffic). Open Street Map[3] also provides map data, in this case open source in XML format, with pre-generated raster tiles of various styles. In both cases an API is provided which is a good basis for projects involving wide-scale located data.

The combination of data sources also introduces various potential problems: Data arising from different survey processes may be produced at different times (creating mis-match where features change), to various scales (implying different accuracy), with different equipment (implying different accuracy and error patterns), using different ontologies to describe common features. Data which are derived from paper maps can also suffer from distortions in the drawing process, designed to make the original data legible, e.g. magnification of roads and slight movement of otherwise overlapping features. The combination can lead to *registration* problems in the combined data set, where features do not align quite as they do on the ground; and where real experience does not quite align to the map data.

The issue of error from Chap. 6 is then raised again: features in the world move; reference systems for describing the world are inaccurate; survey methods have limits on their accuracy and potential for error; systems for describing the conceptual world are varied; and processing of location data can introduce numerical errors.

7.2.2 Points of Reference

Many location systems use coordinates in a flat plane as a reference system to describe location. The plane is an approximation for an area of the earth, which holds true over day-to-day experience within smaller areas, the curvature of the earth being quite gently varying. The mapping of the curved earth to a planar representation is known as a *projection*. The plane makes much of the geometry involved in calculating relationships between points far simpler. However, it should be noted that this is not precise and introduces distortions, depending on the projection used some of: area, shape, direction, bearing, distance and scale become inaccurate to some degree—as illustrated in Fig. 7.1. Note that the final projection, a rectilinear grid is convenient, but only valid as an approximation over relatively small areas. Similarly modelling the earth as a sphere is incorrect when examined in detail (the north-south radius is approximately 21 km shorter than the equatorial radius); an ellipsoid is a better approximation—although there is not simply one choice of ellipsoid as a best fit. The fit is usually determined with reference to the oceans as they have a substantially uniform surface.

Latitude and longitude are used as a reference to position on the spherical or ellipsoidal model, but not exclusively—much location information is described in terms of local grid systems (and most countries or regions have their own approximations which introduce the least error within their own scope), and so transformation is still required.

[3] http://www.openstreetmap.org.

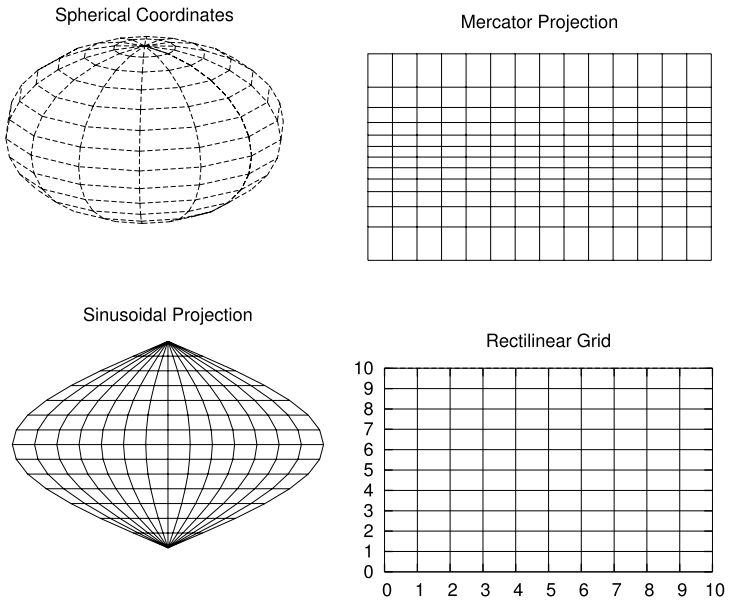

Fig. 7.1 Map projections

For elevation sea level is commonly used as a reference point, but sea level varies with time (due to local volume of water, tides etc.)—so a local mean sea level is usually used. Reference to the centre of gravity, or the intersection of the spin axis and the equatorial plane, might be globally more precise but has (historically at least) been difficult to use in practise. GPS makes a global reference easier to acquire, but elevation is not the most accurate aspect of GPS. Barometric altimeters are still in widespread use (e.g. in aircraft, some cycle computers and GPS units) but are only as accurate as their calibration. To know your height the air pressure at a known elevation is required. To differentiate change in height from change in pressure requires fresh reference data—often from weather stations (landing an aeroplane to check air pressure in order to know your altitude presents a problem!). Radar and sonar methods require knowledge of location and a known reference point to be in range.

It is traditional to align one axis of the grid to the North–South axis. Again, this introduces choices and approximations: The true and magnetic north poles are not exactly aligned and the difference varies over time. The axis of spin also varies in relation to the orbital plane of the earth. The result is that grid coordinates for a known point need to be described with reference to both a projection and a datum for that projection, which describe the interpretation of those coordinates with reference to a current compass reading or stellar reference point.

7.2.3 Conversion Between Coordinate Models

Once data has moved into one coordinate model it can still be transformed into other models, although we should be careful of cumulative errors. However, the calculations involved in coordinate conversion can be complex (certainly requiring proper floating point support) and even iterative—giving some concern about the predictability of performance and the suitability of some devices for making these conversions.

If a location model is being developed which combines inputs from different sensors it is generally preferable to have a single internal representation of location, even an abstract one, and convert inputs to this and outputs from this [15]. This reduces the size of the set of conversion functions (although individual functions can be complex still) and promotes a consistent internal treatment.

7.2.4 Human Descriptions

Unfortunately, most coordinate models are rather abstract and contain few physical manifestations—so human descriptions do not naturally come in the form of grid references. Indeed, even given a reference point most people are not particularly accurate in estimating distance and direction. A more usual spatial description by a person is descriptive:

- A symbolic name, e.g. “my home”. Note that this location is with reference to the person making it.
- An address, e.g. The West Pier, Brighton, UK. As databases of addresses are often available this form is most easily translated to a coordinate system.
- With reference to a landmark, e.g. “the tall tower in the middle of town”. Note that which “town” is implicit in this example.
- Relative to some known location, e.g. “next door to my office” or “1 km from here along this road”.

Variously things which are *adjacent* or *visible from*, things which have a *visible connection* or *relationship*, things which are *labelled* or *match some pattern* and things which are *contained*. A common technique in navigation (in particular in orienteering) is to aim approximately towards a goal with the intention of finding a large or linear feature—this does not require accurate aim or distance measurement. Having found the reference feature, e.g. a track, possibly deliberately *aiming off* to one side, the real target can be found in relation to this. In spatial databases (see [19] for a deeper discussion of this) this generates a requirement that such relationships can be described as search terms. When collecting data in pervasive systems which are indexed by location we should be aware of the need to use that data later and the value of relationships between data as well as absolute positioning of data. However, the literature generally defers to coordinate systems and/or GIS

approaches to achieve this. The main departure from this is in some sensor systems where location models are developed by using radio visibility between nodes.

Having established sources of location data and coordinate systems it is necessary to be able to interpret the location. For a map to display "you are here" coordinates are sufficient (indeed, preferred). For more complex behaviours it is the nature of the place, rather than the particular place in the world, that is often of most interest, e.g. being at home, at work, in my office, in your office, in a lecture theatre and so on.

In Leonhardt and Magee [14, 15] inputs which describe the located objects within *cells* are considered; these include an active badge system, UNIX `rusers` login detection, and GPS. Cells naturally describe objects—as discussed there is always an error associated with a location reading, and most objects occupy a measurable volume in space. A cell can be described as disjoint, identical, within, overlapping or adjacent-to another cell. The cells arising from active badge detectors and logins have well defined zones of meaning—the space described is a volume not a point. Cells derived from GPS are regions on a coordinate space, which have a variable size due to reception quality. In general there is a mapping between the various location sources' coordinate systems, via the cells—although only to the granularity of the cell. An illustration of four cells is given in Fig. 7.2. Here cells A and B are disjoint in space, and cells C and D are contained within cell B. Clearly describing all possible GPS locations with all possible accuracies in advance is impractical, so cells are generated dynamically to represent such readings. In this case a mapping from a coordinate space into the higher level model is required. Given the spatial descriptions of locations a hierarchical model of *location domains* is built up: a university containing a campus, containing a building, containing a room etc. Another model of location, which uses a similar semantic map and operator arrangement, is described with a variety of applications in [3].

In using models of the world based on GIS vector data the semantics of the data are given: each feature has a separate data entry, which includes both the coordinates and meta data. Key to the latter is a pointer to a type system or ontology describing the possible kinds of feature. For some data this will be equivalent to a map legend, for others GML and OWL and similar will be used.

7.2.5 Relationships

It is all very well being able to describe individual objects in space, but many of the uses of location context we identified above relied upon being able to describe relative location:

- Who is near?
- Is w within 100 m of me? (which requires finding objects within a scope)
- Where is the nearest x? (which requires measuring distance between objects)
- How do I get to z? (which requires finding a connected path through a series of transport paths)

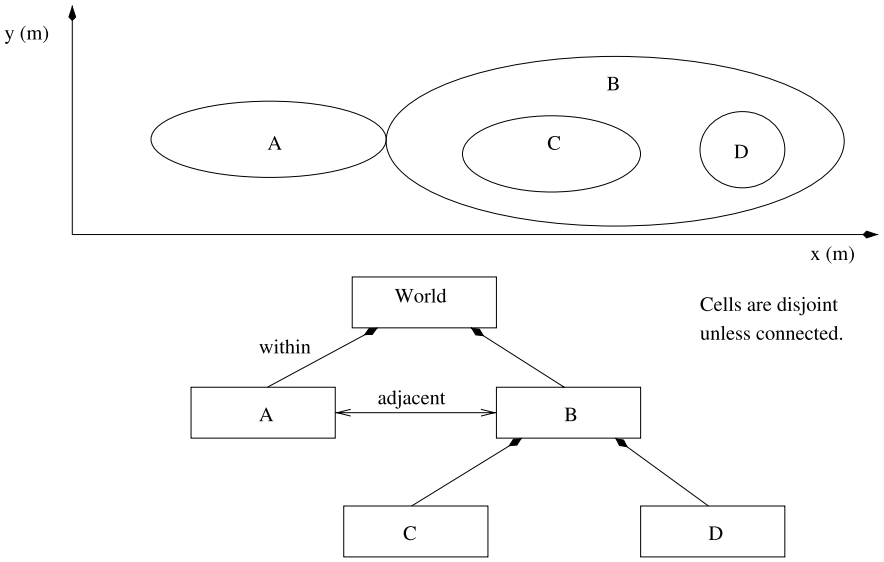

Fig. 7.2 Geometric and symbolic (cell) locations, based on [15]. Ulf Leonhardt and Jeff Magee. Multi-sensor location tracking. In *4th ACM/IEEE Conference on Mobile Computing and Networks (MobiCom)*, pages 203–214. ACM, 1998, http://doi.acm.org/10.1145/288235.288291 © 1998 Association for Computing Machinery, Inc. Reprinted by permission

A useful working model of object comparisons is given in [5] and summarised here and in Fig. 7.3. An object (shown as ellipses, but applicable to any polygon with a boundary) has three zones in space: exterior, boundary (∂) and interior ($\circ$). The intersection of two objects may be considered by comparing each of the zones of each of the objects, or by looking at four spatial relationships: intersecting boundaries ($\partial\partial$), common interiors ($\circ\circ$), boundary as part of interior ($\partial\circ$) and interior as part of boundary ($\circ\partial$), as shown in Table 7.1. This second representation avoids reasoning about the exterior of an object, which can simplify complexity where there are many objects. Each of these four relations can be empty ($\emptyset$) or non-empty ($\neg\emptyset$).

The GIS / spatial database literature, indeed even the cited paper, contains other aspects of relationships which might need to be considered, e.g. whether objects meet at a point or along a length and the handling of relationships between collections where multiple points, or polygons with "holes" in must be considered.

7.2.6 Summary

The processing model for context in Fig. 5.1 remains appropriate here: location data requires sensor processing and a conversion into a low-level model. The low level data can be further processed to give a high level model. Location sources can be fused, at high or low level, to provide greater accuracy or confidence. Location can

Table 7.1 A mathematical model of topological relationships based on [5]

Relation	$\partial\partial$	$\circ\circ$	$\partial\circ$	$\circ\partial$
disjoint	Ø	Ø	Ø	Ø
equal	¬Ø	¬Ø	Ø	Ø
meet	¬Ø	Ø	Ø	Ø
contains	Ø	¬Ø	Ø	¬Ø
inside	Ø	¬Ø	¬Ø	Ø
covers	¬Ø	¬Ø	Ø	¬Ø
covered-by	¬Ø	¬Ø	¬Ø	Ø
overlaps	¬Ø	¬Ø	¬Ø	¬Ø

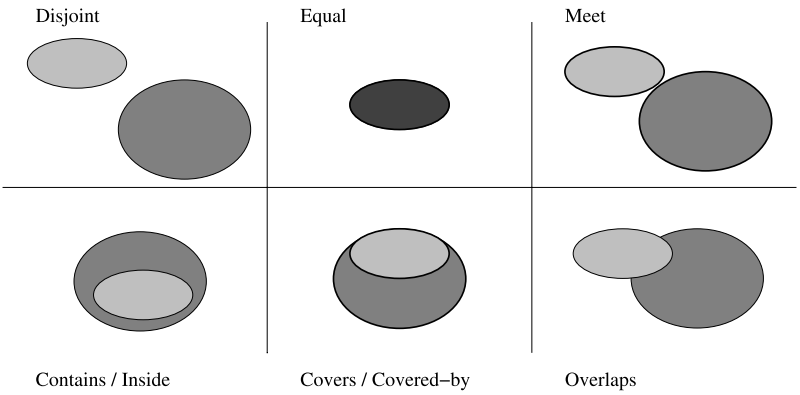

Fig. 7.3 A visual model of topological relationships based on [5]

be described as a state, or that state can be interpreted by an event detector. A similar location stack, with fusion, is also described in many papers, e.g. [10, 15].

In [11] Jiang and Steenkiste discuss coordinate systems, and take the view that neither coordinate models nor semantic models by themselves are satisfactory. They propose a URI system for describing places (volumes) in either, or a mix, of these systems. The mix of description systems also allows for conversion between systems with local, rather than global, points of reference. This is integrated into a database system to provide for efficient searches in a location space (see also below on R-Trees), with the usual spatial relations.

7.3 Sources of Location Data

Location sources can take various forms, but can be classified into two basic forms: those which determine location with respect to global coordinates, using known reference points; and those which simply determine location with respect to other

nodes in the system. We shall be concentrating on the first sort here. There are various techniques for calculating the relationship of a node to the nodes which have a known location, which we shall survey below. We focus on those where the reference nodes are easily found, and so are most applicable to general purpose pervasive computing. The more general solutions for fewer known reference points or purely relative location are typically addressed in texts on sensor networks. These various sources have different properties: some require particular antennae, membership of services, certain processing power; some are sensitive to configuration, environmental factors and require up-to-date survey data; some drain power faster than others or provide information faster than others; some are more accurate than others; some describe their data in different ways; each has a range over which it can operate due to the deployment of its associated infrastructure. Between the various systems a location of some quality can often be constructed. We shall now briefly survey the issues and highlight the main choices to examine when considering a particular technology. Each can be classified in multiple dimensions, including:

1. What reference system is required to interpret the location data
2. Whether locations are described as coordinates or symbolic
3. Deployment and maintenance costs for infrastructure and marginal deployment costs for additional located objects
4. Resolution of location information
5. Error characteristics of location information
6. Coverage, including scale of operation and environments that can or cannot be located-in
7. Capacity for tracking additional objects
8. Whether location is computed on the located object or by some central service

You may wish to consider some of these (and possibly other) criteria as you read this, and research more detail on location systems.

7.3.1 Cellular Systems

A primary source of location information is gained through proximity to a sensor. The range of the sensor may vary, but if presence is detected then the located object can be said to be within that sensor's range (or cell). In some cases this requires contact with the sensor: pressure on a seat etc. One removed from this is contact (or other very close connection) with some proxy for the located object, e.g. terminal activity on a computer, and swipe cards on doors as proxies for user location, RFID tags as proxies for object location.

If cells are designed such that there is *no* overlap, then cell ID can translate to a region in a trivial manner. Note that the cell may broadcast its ID, or a device may broadcast a query for location information. In a system where a device will only see or be seen by one reference node (or beacon, base station) there is little difference, apart from any overheads.

As cells become more dense, so that regions usually overlap, the location can become more accurate as the intersection of cells reduces the possible region referred to. If the cells are defined as regions around the known location of reference nodes in a regular grid then the average coordinate in each axis of the reference nodes seen at some point defines the centroid of the device to be located. As the overlap ratio of beacon range to separation increases the average error in this centroid becomes a smaller multiple of the beacon separation [4]. In real world settings the variation in beacon position and range due to the environment means that a regular grid may not be seen, but geometric intersection of known (or expected) signal patterns can still be used to identify a region which contains the device to be located. Where cells overlap there is also a greater possibility of beacons or query responses overlapping, although being static beacon timing may be easier to plan than the response to a query from a device with an unknown location.

Location systems can be derived from ad-hoc sensors. In this category we include cellular systems from 802.11 SSIDs gained by "war-driving" and Bluetooth beacon sensing. We note these as being different as (in the general case) the number of beacons can be of the same order as the number of users; the ownership of beacons is widely dispersed (certainly more than single infrastructure owner or a few mobile phone operators); and because the operation and location of individual beacons is variable. In the case of Bluetooth beacon sensing between mobile phones all that can really be established without further data is that the two devices are co-located within a mobile cell (*cell* as used above not the phone operator's base station derived cell) which just encompasses the two phones by the range of Bluetooth. The Place Lab project[4] [13] uses 802.11 and GSM base-stations as beacons and demonstrates very good coverage in somewhat urban areas and the ability to operate both indoors and out. The paper includes a useful discussion of the trade-offs inherent in designing location systems, in this case some accuracy is lost in allowing easy set-up, good coverage and a real consideration of privacy issues.

7.3.2 Multi-Reference Point Systems

Some location systems require inputs from multiple sources in order to arrive at a location, by *triangulation* or *trilateration* e.g. GPS, ultrasonic systems with a mesh of sensors in the room, use of signal strength from multiple wireless network base-stations etc. Here geometry is applied so that multiple distance (trilateration) or angle measurements (triangulation) from beacons which do not map into a single plane are combined to find the place of intersection. In general using distance measurements requires (at least) $D + 1$ measurements, i.e. 3 measurements for 2D, 4 measurements for 3D. Where the system implies extra information about the relative location of object of interest and transmitters is known this number can be reduced, e.g. in GPS satellites are above the receiver. Using angular measurements is similar,

[4] http://www.placelab.org.

but the distance separating two sources is usually required; sometimes a constant reference such as magnetic north is used. In both cases installation specific knowledge can be applied to simplify the problem. Distance is rarely measured directly, with a tape measure, but through other measures such as time of flight and signal attenuation. These measures can sometimes be had with little cost due to existing infrastructure, e.g. 802.11 base stations, but can require calibration to the local environment. Time of flight either requires precisely synchronised clocks (rather hard to achieve in practise); or two signals, commonly RF and audio, so that time of flight is determined from the difference. Over large scales Vincenty's formula should be applied for calculating distance on an ellipsoid.

We shall not discuss all triangulation systems here, although examples of ultrasonic, wireless LAN, cellular phone network, wireless with purpose designed infrastructure and others can be found in the literature. By way of example we shall give a brief review of one of the most common triangulation systems: the Global Positioning System (GPS).

7.3.2.1 Global Positioning System

GPS is a US military system for providing location through measuring delay (and hence distance) from multiple satellites—a form of trilateration. While it was deployed as a joint-forces project it has been made available for civilian use as a common good. It uses highly accurate atomic clocks on satellites to transmit a time signal, which also send data on their orbits ("ephemeris") to allow calculation of time of flight and approximate data on other satellite positions to aid signal acquisition ("almanac"). At least four satellites must be visible to achieve an accurate 3D location plus time, from a constellation of at least 24 satellites in medium Earth orbit. This orbit means that each satellite moves across the sky when seen from a stationary point on the Earth's surface, requiring a receiver to scan for new signals but providing robustness against individual satellite failure and ensuring satellites are distributed across the sky to facilitate global coverage and accuracy.

A message from an individual satellite is encoded using Code Division Multiple Access (CDMA), allowing all satellites to share a frequency band. This message is decoded and arrival time recorded using an internal clock at the receiver. This clock is not synchronised with the satellite clock, so cannot by itself compute a distance. However, the difference in time of flight between satellites together with ephemeris can be used to compute an initial distance estimate. Due to the very short times small timing errors imply large positioning errors and the distance calculation must include corrections for relativistic effects. This distance describes a sphere in space centred on the satellite (although it is assumed the receiver is below the orbit, so a full sphere is not required). As additional satellite signals are decoded additional spheres can be described. Receiver position is computed from the intersection of the spheres. This requires that the satellites are not lying on the same plane as each other; but their number and orbits help to ensure this condition is met. A fourth satellite signal is usually used to correct the GPS receiver's internal clock,

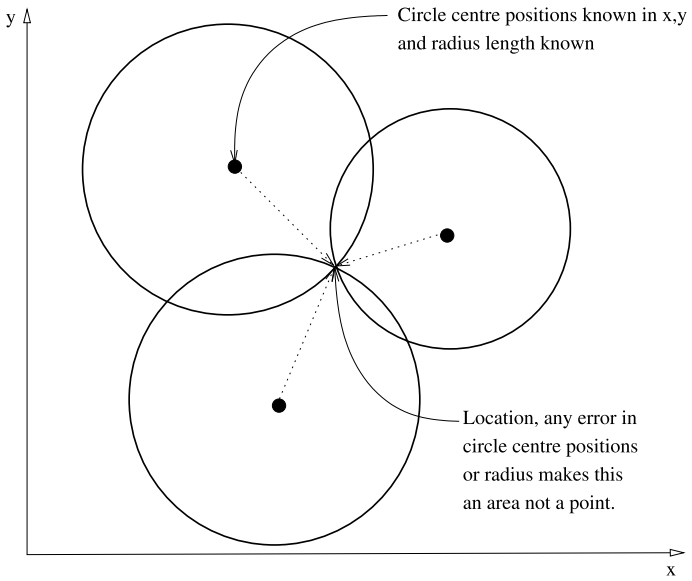

Fig. 7.4 A 2D illustration of lateration, with circles of known centre and radius

by measuring the distance from it's sphere to the 3-satellite position estimate. With an accurate clock a new estimate of time of flight, and hence distance to satellite, can be produced. This computation is typically an iterative estimation, improving the local clock accuracy and acquiring additional signals to refine the accuracy of the distance estimates and hence reducing the error in the position estimate. During initial signal acquisition movement makes this refinement of timing rather difficult and leads to delays in achieving an accurate estimate of location. The principle of lateration used in GPS is illustrated in Fig. 7.4.

There is some inherent uncertainty in GPS data, arising from various factors including:

- Satellite clock accuracy causing drift over time, this is frequently updated.
- Accuracy of the measured signal delay limiting resolution.
- Accuracy of the model of real satellite position for calculating signal delay, this is frequently updated.
- Atmospheric variation, which is the least easy to model and predict the effect on signal time of flight.
- Selective availability, introducing deliberate errors of up to 100 m into the signal for military purposes, can be corrected for using differential GPS.

This is not the only satellite based positioning system, but it is the most widely known and GPS receivers are easily available consumer products, built into many smart-phones and other devices. The most obvious alternative system is the EU "Galileo" system whose satellites are currently being deployed.

We previously identified several categorisation criteria for location systems. We shall conclude with a note of GPS' properties in these:

1. The reference system is with respect to a datum for a global model.
2. Locations are on a coordinate system.
3. Deployment and maintenance costs for infrastructure: satellite deployment and operation is very expensive; satellites have a finite lifetime (approx. 15–20 years). Marginal deployment costs for additional located objects: GPS receiver circuits are consumer items and cheap, hand-held dedicated GPS units and mobile phones containing GPS can be bought for around £100.
4. Resolution of location information: 5–50 m depending on conditions, can be improved with additional infrastructure.
5. Error characteristics of location information: depends on number of satellites visible, atmospheric conditions etc. as discussed above. Generally a well understood error pattern, refresh of location data provides an estimate of current accuracy through variation in reported location.
6. Coverage: global, outdoors.
7. Capacity for tracking additional objects: unlimited, no infrastructure costs.
8. The location is computed on the receiver, with no central registration or awareness of connection.

7.3.3 Tagging

Systems with *tags* allow location to be determined in two ways:

- By fixing tags at known locations, when a tag is scanned the scanner must be within scanning range (which depends on the technology) of the tag's location.
- By having tags report to some infrastructure cellular or triangulation techniques can be applied.

The latter approach featured in the active badge project [21] and is effectively a variation on the schemes elsewhere in this chapter. In this section we shall discuss the first case—where scanning a tag generates a location event describing proximity to the tag. This type of tagging is common in what is known as "the Internet of Things", where everyday objects have an internet presence (typically a web presence), which is sometimes deployed in a manner overlapping with augmented reality. In some cases this presence will be achieved by embedding computation and networking into the object in other cases there will simply be a presence about the object on the web. In either situation, it is common to "tag" objects to facilitate identification of their address in the internet, and visual tags can facilitate user-identification of objects of interest. We can further consider five classifications axes of this approach:

1. The "location" of the tag may be fixed, e.g. tags to identify rooms, bus stops etc., or the tag may be attached to a mobile object, in which case the location *only* describes proximity rather than absolute position.

2. The tags may be unique, e.g. Bluetooth tags with a MAC address, the tag may have duplicates, e.g. identifying a shop by brand not location, or a given location may have multiple tags, e.g. at different entrances or identifying different properties of the place. Where tags are intended to be unique one should consider whether security or functionality issues arise from forgery or accidental reuse or tag re-positioning.
3. The range that a reader can read a tag over. This may be variable for a given technology, influenced by tag and reader design, available power, and the environment.
4. The cost and integration involved in deploying the tags and the cost and integration involved in reading tags. There may be set-up costs for a system where tags refer to an infrastructure.
5. The behaviour mode of the tag—reader system, including: whether the reading of a tag is automatic or manually triggered; and whether the reading is an entry or trigger event or a presence state.

The first classification is generally a property of deployment rather than technology, as the usual model is that the marginal costs for tags is very low (often to the point of being disposable) and the maintenance of tags is very low (often ruling out powered tags). This cost model is an important property for the Internet of Things model (and pervasive computing generally), where everyday objects would acquire a tag and so quantities of tags are very large compared to quantities of readers, and the functionality tends to be concentrated towards the user rather than the environment. There are various systems which can function in this way, we shall briefly examine three which give rise to systems with different properties: Bluetooth, RFID and QR codes.

7.3.3.1 Bluetooth

Bluetooth enabled devices can be used as a form of tagging. To act as a tag a device needs to have its Bluetooth switched on and discoverable. To act as a reader a device needs to scan for other Bluetooth devices. The tag data can simply be the tag's unique Bluetooth address, which tells the reader little about the device but is easy to acquire and could be correlated with other data. In this case the tag is not aware of the reader detecting it and where tags are in fact personal devices (quite common) it may give rise to privacy concerns—particularly where the device has been given a personal name, or where the readers store scans and correlate with other data (a network of readers in other locations, very particular reader locations, video observations etc.). Intrusive observation can be defeated by making personal devices not-discoverable, except when pairing and by limiting correlation of reading events with other data.

In terms of our four distinguishing properties:

1. The tags are generally unique, but a given location may have multiple tags.

2. The range depends on power and antenna design for both reader and tag. A typical configuration would allow detection over 5–10 m, although longer range devices may extend this and substantial walls and metal structures may reduce it.
3. Bluetooth circuits are built into many consumer devices, where these already exist the cost approaches zero. The cost for additional tags is low. The energy cost of tag beacons and reader scanning can be quite high, cf mobile phone battery life with Bluetooth on.
4. Once switched on and properly coded the scanning can be automatic, frequent, registers presence and is invisible to the human observer—although each of these properties can be modified in code and/or device configuration.

7.3.3.2 Radio Frequency ID

Radio Frequency ID (RFID) tags are widely used in warehouse, retail, library and transport situations, supporting theft detection, stock identification and tracking and account identity for payment. Reading is contact-less, using radio (unlike a smart-card), with range depending on available power and antenna design—typically a few cm to a metre. A tag consists of an aerial and some small memory and/or computing capacity plus two-way radio interface circuitry, typically printed on a label or embedded in a card. A wide range of radio frequencies and protocols (some standard, some proprietary) are in use, with choice affecting tag and reader size, power consumption, range and data transfer speed. Different systems also have differing abilities to handle reading multiple in-range tags at once. The tag and aerial are often several cm across, but miniaturisation is progressing, antenna design for a desired range being a limiting factor. It is possible to embed a power source into a tag, but we will focus on the more common (for this application domain) passive-RFID, where power is extracted from the radio signal of the reader in order to power the tag while it is being read. A few mobile phones have built-in readers, but in general the hardware is special-purpose. The tag data depends on the design of the tag, and can range from a simple numeric ID (used in a similar way to a bar-code for stock control), to more complex structured records. The tag computation may range from unauthenticated, un-encrypted data exchange on demand, to protocols which require an identifier and provide simple encryption; some protocols may also record tag reading events or cause tag data to be changed. The modification of tag data increases the need for security provision but allows the tag system to function where access to a database indexed by tag data is problematic. The main barrier to invisible tag reading is the need to camouflage long range antennas, and as possession of tagged objects could reveal quite personal data their use has been somewhat limited.

1. The tags may be unique where suitable security provisions have been made but for simpler technology should be treated as duplicatable, a given location may have multiple tags and it may be possible to read them all in one scan.
2. Tag range is from mm to a few meters, generally requiring clear air.
3. The marginal cost of a tag is very low and they are widely used as disposable labels for low value goods. The costs of tag producing / writing hardware is higher.

Reader costs are moderate, often requiring integration with other systems. Power costs of reading are significant, but passive RFID has no routine maintenance cost.
4. Once switched on and properly coded the scanning is automatic, frequent, registers presence and is invisible to the human observer. Modifying this behaviour is hard as the tags and scanner are less likely to be easily user programmable and incorporate displays than in the Bluetooth case.

7.3.3.3 Bar Codes

Bar codes are commonly found on printed product packaging, and provide a simple optical ID system. We shall focus on 2D matrix bar codes, in particular QR codes[5] here as these are widely used and easily read with mobile phone cameras. The tag can encode numbers, alphanumeric codes, 8-bit binary or kanji, the typical internet-of-things approach would be to encode a URL, but the data is arbitrary. The code can be read from any angle, is somewhat damage resistant, and the detail required depends on the amount of data being encoded—so short URLs are preferred. To deploy codes requires some software to encode the data into a black and white image (widely and freely available) and a printer or display. To read the code requires a simple camera (mobile phone cameras are ample) and software—again, widely available and increasingly supplied with the phone. The reading of such a code is a one way process, the tag being unaware unless it is displayed by a device which is informed of any data access triggered by the encoded data. Lighting conditions (bright lights and darkness) can interfere with tag reading, particularly with glossy weatherproofing. Unless a camera is concealed the act of reading a tag is generally visible.

1. The tags are easily duplicated and a given location may have multiple tags, if there is space to display them.
2. The range depends on lighting, printer size and camera quality, but is usually under a metre in practical applications.
3. Cameras are built into many consumer devices, where these already exist the cost of reading approaches zero. The monetary and power cost for additional tags is low for printed tags, higher for active displays.
4. Reading a tag is typically manually triggered, as the device often has other purposes and required re-enabling after a read, but this behaviour could be changed. The reading of a tag is typically an event.

7.3.4 Fingerprints and Scenes

A final location system we shall mention only briefly is where locations are recognised through reference to a database of previously surveyed features. Commercial

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

and free systems are available which use this technique, in particular with 802.11 wireless networks. Network base stations can be used to develop a cellular system, where cells are defined by the coverage of a base station. As wireless networks have become more widely deployed it is easy to find that multiple base stations are "seen". If the location of each base station is known then the centroid of these can give an approximation to the receiver's location. If the networks are operating in an open space then trilateration based on received signal strength can be used to find the location. However, for the general case trilateration is problematic as antenna properties may vary and obstacles cause complex changes to signal propagation. It is in this situation that fingerprinting is used.

A survey is made of the space to be covered, by recording received base station identifiers and, ideally, signal strengths and correlating these with some external location system, typically manual registration on a map or by using GPS. This process can occur with high resolution, e.g. on a 1 m grid in an office space, e.g. Ekahau[6] or by driving through a built-up area, e.g. [13]. For high resolution signal strength is required, and it may be necessary to take readings for a given point with people in different places relative to the receiver aerial to form a reliable model. For all uses periodic re-surveying is required as objects which affect signal propagation change and deployment of base stations changes (through user switching at different times of day, renaming, replacement and addition / removal). To use the system to locate an object at a later date the wireless network is scanned and base station IDs and signal strengths used to query a database. The database will report a best match taking account of noise and variation models, possibly interpolating between reference points.

Related to this technique is *scene analysis*, which is a visual technique but also operates by comparing a reading of the location with a database of previously recorded data. Here photographs are used to build models of prominent features in a located database. To use the system to locate an object a photograph is taken. The picture is analysed for key features, e.g. buildings, hills etc. These are then compared to the database, taking into account changes due to camera lens (and so distortions and relative sizes of features) and angle of view. Even for a restricted coverage this is a complex visual search, but the subject of research in image analysis and information retrieval communities.

7.3.5 Summary

As we have seen, there are many ways to compute location and an even greater number of technologies applying these principles. Different technologies give different trade-offs in the classification we gave at the start of this section. We shall now review these categories, noting systems which occupy various places in it.

[6] http://www.ekahau.com/.

1. Many systems, including GPS and Place Lab, use a global *reference system*. The Active Badge system describes a location relative to a smaller coordinate space, defined by the system coverage.
2. *Coordinate* systems include GPS and within-room positioning systems. *Symbolic* locations are given by most tagging systems. Other systems, such as fingerprinting and cellular systems may be designed to use either approach.
3. As discussed above deployment and maintenance *costs* for GPS infrastructure is large, while QR-code tagging can be very cheap. Costs associated with locating an object are often absorbed into other aspects of the system, such as wireless networks and cameras in mobile phones.
4. *Resolution* varies from sub-cm for some RFID (although limited to proximity to a tag), through 0.1 to 5 m for many indoor systems; to 5 to 50 m for GPS and $<$150 to $<$300 m for different forms of E911 legislated phone location.
5. *Error characteristics* of location information are generally very complex. The simplest are defined as ranges around tagging systems. Stability of any infrastructure is a significant factor.
6. GPS gives global *coverage* but requires something close to out-of-doors with clear sky on the Earth; ultrasonics require in-building infrastructure and only operate within rooms where the infrastructure is deployed. Tags create cells of location coverage.
7. Capacity for tracking *additional objects* is very high in many of these systems, but cellular systems are generally limited by base station capacity. The willingness of owners to allow objects to be tagged (and physical space) limits QR codes' capacity for located objects.
8. Most of the systems we have examined compute location at the receiver, as this simplifies reliability and scalability; the active badge and use of any central reference databases in tagging systems and fingerprinting are obvious exceptions. Privacy should be considered separately as once computed it is easy to reveal location information through an application even if it is computed locally.

High resolution, global, indoor and outdoor, scalable, reliable and cheap coverage has not yet been achieved. To obtain even reliable, global coverage requires a system which can combine location data from different sources—converting between reference and coordinate systems and adapting to varying resolution, error characteristics. Where the object to be located is covered by multiple location systems at once, the data from these systems can still usefully be combined. Depending on the reference and coordinate systems we may achieve some of:

- Better reliability
- Greater accuracy and/or reduction of error estimates
- Greater flexibility to use either coordinate or symbolic locations

The needs of the system at hand are, as ever, crucial in selecting a trade-off between these factors and the modes of use required.

Having obtained this data it will typically be used to support location awareness in applications. In most cases this implies that data is stored with a location index. This is the subject of the next section.

7.4 Storing and Searching Location Data

Storage of data for efficient look-up using a coordinate data type requires data structures optimised for this sort of use—which is not necessarily the same as for more traditional numeric data types. As discussed in Sect. 7.2 we must consider: multi-dimensional (2 or 3-D) data, relations other than $==, >, <$ and data which may occupy a space rather than a single value. However, if we consider the storage as a database, then usual drivers remain present: efficient selection queries, insertion and index update; and joins with other data. When using spatial data algorithms and structures which allow approximations to quickly limit the set of data of interest, and focus the more complex computation around these operations on a limited data set—in the same way that e.g. B-tree indices are used for other data types.

In this section we give a brief overview of R-Trees and spatial relations, which tackle the main issues and are the basis of many of the techniques in mainstream use.

7.4.1 R-Trees

A data structure which is widely used in processing geographic data is the R-Tree [7], and its variants, e.g. R*-tree [2]. An R-Tree is a tree data structure which uses location to organise grouping of data within nodes and the relationships between parent and child nodes. Each tree node contains data for a region in space, known as a *bounding box* (we shall describe a 2D model here with rectangular boxes, but n-dimensions are possible). The bounding box minimally contains all the located records in a leaf node or all the child bounding boxes in a non-leaf node. The bounding box for a node may overlap with the bounding boxes for other nodes. The bounding box is a minimal description, so that addition of new data can change the bounding box for a given node. Nodes have a bounded capacity for data items (a configuration parameter), so the tree grows by node division where the algorithm for selecting which sub-tree a data item goes into is a crucial factor in the efficiency of the tree. This bounding capacity is both a minimum and a maximum number of records (except the root). For non-leaf nodes these records will be children; for leaf nodes these records will be data points. The root may either be a leaf or a non-leaf node, in the latter case it will have at least two and at most the maximum capacity of child nodes. See Fig. 7.5 for an example—note that the whole possible space does not need to be under any tree node, that actual data items are only held at the leaf nodes, and that nodes do not need to fill their capacity for data items. We also see that the R-Tree is depth-balanced, so that search complexity is uniform, rather than developing hot-spots—all leaf nodes are on the same level.

The R-Tree algorithm is summarised in Listings 7.1 (insertion basics), 7.2 (search), and 7.3 (tree growth) as an explanation—for implementation refer to the detailed presentation in the paper for the R-Tree [7] or one of its variants. We see that the R-Tree structure lends itself to top-down recursive processing. Some detail is omitted, but the general process for most R-Tree variants is illustrated: find the

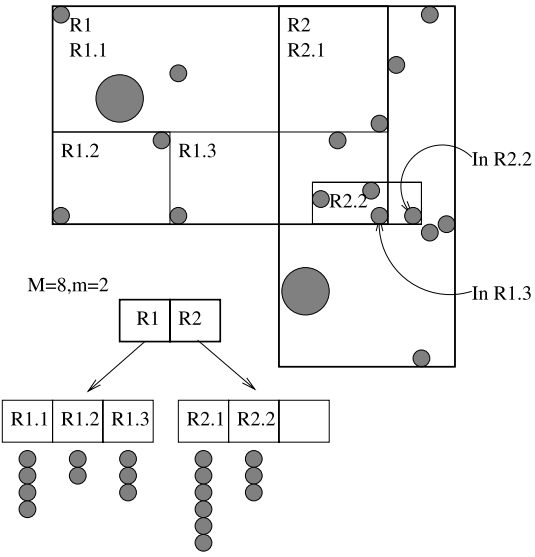

Fig. 7.5 An example R-tree based on [7]. Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In Beatrice Yormark, editor, *SIGMOD Conference*, pages 47–57. ACM Press, 1984, http://doi.acm.org/10.1145/971697.602266 © 1984 Association for Computing Machinery, Inc. Reprinted by permission

most suitable sub-tree for insertion; manage node size; split nodes if necessary. The insertion of data is where most of the possible optimisations are to be found—even the original paper considers three algorithms here. Particular variants of the algorithm have been constructed to improve efficiency for particular circumstances, e.g. static data sets, highly dynamic data sets, and various underlying storage architectures.

Searches across the resulting data structure are driven by data contained in the tree, not by an arbitrary organisation of the space; the index is dynamic and doesn't require periodic rebuilding or any domain knowledge to define. Because nodes may have overlapping bounding boxes a search may visit multiple leaves.

7.4.2 Spatial Relations

Spatial relations are discussed in many papers on location models, including [3, 11, 15, 17, 18]. R-Trees explicitly focus on a *containment* (or *within*) relation: finding located objects contained within a region. Some of the other relations which consistently appear in the literature include:

1. Overlaps: where two regions intersect. Containment can be viewed as a special case of overlaps, but really has different implications. Overlaps can be described as a Boolean, or can give a degree—which describes the proportion of one region which overlaps the other.
2. Adjacent, which can be described as two regions which just meet without sharing any space.
3. Disjoint: where two regions neither intersect nor are adjacent.

```
insert(Entry e)
  l = chooseLeaf(e)
  if (l.size < l.capacity)
    then l.add(e)
    else
      (l,ll) = splitNode(l,e)
      adjustTree(l,ll)
      if adjustTree caused root split
        then create root with (l,ll) as entries

Node chooseLeaf(Entry e)
  n = root
  while (n not a leaf)
   choose n' from entries in n such that:
     n'.bbox needs least enlargement to include e
     if tied, pick smallest n'.bbox
   n = n'
  return n
}
```

Listing 7.1 The R-tree insert algorithm: basics, based on [7]

```
Entry[] search(Rectangle s)
  n = root
  result = []
  while (n not a leaf)
    for all children c in n
      if overlaps(c.bbox,s)
        then result += search c
  // n is a leaf
  for all entries e in n
    if overlaps(e.bbox,s)
      then result += e
  return result
```

Listing 7.2 The R-tree search algorithm, based on [7]

4. Near or close, which are commonly used without qualification in natural language, allowing nearness to be interpreted relative to size and/or speed; and possibly to have a degree of nearness (relating distance to size and/or speed).
5. Distance describes the separation between two regions. Different interpretations exist, describing maximum, minimum, separation of centres or some combination of these.
6. Relative size similarly compares the areas of two regions.
7. Connected, where there is a path from one region to another through a series of other located objects in the space which overlap or are adjacent to each other in

```
adjustTree(Node l, Node ll)
  while (l != root)
    update bbox entry for l in l.parent
    if (ll != null) // split occurred
      then if (l.parent.size < l.parent.capacity)
        then create entry in l.parent for ll
          // pointer to ll + bbox for ll
      l = l.parent, ll = null
      else (l,ll) = splitNode (l.parent, ll)
    else l = l.parent, ll = null

(Node,Node) splitNode(Node l, Entry e)
  n1, n2 = new Node
  (s1, s2) = pickSeeds(l)
  n1.add(s1), n2.add(s2)
  l.remove(s1), l.remove(s2)
  for each e' in l.entries + e
    if just one of n.size < n.minSize
      then assign remaining entries to n; break
    add e' to n1 or n2 such that:
      n.bbox needs least enlargement to include e'
      if tied pick the smallest n.bbox
      if tied pick the smallest n.size
      if tied pick either
  return (n1,n2)

(Node,Node) pickSeeds(Node l)
  in each dimension find e in l whose bbox have:
    highest low side, lowest high side,
    highest high side, lowest low side
    sep = highest low side - lowest high side
    size = highest high side - lowest low side
    norm_sep[dim] = sep / size
  return entry pair with largest norm_sep
```

Listing 7.3 The R-tree insert algorithm: tree growth, based on [7]

a chain. It may be desirable to constrain the semantics of the features forming such a chain, e.g. to road.

7.5 Summary

As with other forms of context, we should strive to understand the behaviour of the data we are working with, its true meaning (and how this may differ from its apparent meaning), and its error modes. With location sensing the state of the art is well developed: sensor systems are varied and widely deployed; data are available to

describe the world in great detail, both conceptually and positionally; algorithms to manipulate sensor data and existing information are mature. Despite this, location remains a subject of active research: new sensors continue to be devised; greater accuracy and efficiency are desired; lower cost and power consumption are required for truly pervasive deployment; and the problem of human meaning and human needs and behaviour is highly complex.

7.6 Suggested Readings

In this chapter we have covered and expanded upon several topics. You may wish to choose readings from several of the following topics, or to pick one and explore more deeply.

On GIS and the management of spatial data:

- Shashi Shekhar and Sanjay Chawla. *Spatial Databases—A Tour*. Pearson Education, Prentice Hall, 2003

On location sensing:

- Jeffrey Hightower and Gaetano Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, 2001 (and also the extended technical report UW-CSE 01-08-03).
- Eamonn O'Neill, Vassilis Kostakos, Tim Kindberg, Ava Fatah gen. Schieck, Alan Penn, Danae Stanton Fraser, and Tim Jones. Instrumenting the city: Developing methods for observing and understanding the digital cityscape. In Paul Dourish and Adrian Friday, editors, *Ubicomp*, volume 4206 of *Lecture Notes in Computer Science*, pages 315–332. Springer, 2006
- Anthony LaMarca, Yatin Chawathe, Sunny Consolvo, Jeffrey Hightower, Ian E. Smith, James Scott, Timothy Sohn, James Howard, Jeff Hughes, Fred Potter, Jason Tabert, Pauline Powledge, Gaetano Borriello, and Bill N. Schilit. Place lab: Device positioning using radio beacons in the wild. In Hans-Werner Gellersen, Roy Want, and Albrecht Schmidt, editors, *Pervasive*, volume 3468 of *Lecture Notes in Computer Science*, pages 116–133. Springer, 2005

A special issue on labelling the world in IEEE Pervasive Computing magazine provides a range of approaches to tying location to places Tim Kindberg, Thomas Pederson, and Rahul Sukthankar. Guest editors' introduction: Labeling the world. *IEEE Pervasive Computing*, 9(2):8–10, 2010

On location models:

- Ulf Leonhardt and Jeff Magee. Towards a general location service for mobile environments. In *Third IEEE Workshop on Services in Distributed and Networked Environments*, pages 43–50. IEEE, 1996
- Changhao Jiang and Peter Steenkiste. A hybrid location model with a computable location identifier for ubiquitous computing. In Gaetano Borriello and Lars Erik Holmquist, editors, *Ubicomp*, volume 2498 of *Lecture Notes in Computer Science*, pages 246–263. Springer, 2002

- Christian Becker and Frank Dürr. On location models for ubiquitous computing. *Personal and Ubiquitous Computing*, 9(1):20–31, 2005

On *R*-Trees and R^*-Trees:

- Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In Beatrice Yormark, editor, *SIGMOD Conference*, pages 47–57. ACM Press, 1984
- Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R*-Tree: An efficient and robust access method for points and rectangles. In Hector Garcia-Molina and H. V. Jagadish, editors, *SIGMOD Conference*, pages 322–331. ACM Press, 1990

On applications of location information:

- Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10(1):91–102, 1992
- Michael Beigl, Tobias Zimmer, and Christian Decker. A location model for communicating and processing of context. *Personal and Ubiquitous Computing*, 6(5/6):341–357, 2002

On error in location data and models:

- Ulf Leonhardt and Jeff Magee. Multi-sensor location tracking. In *4th ACM/IEEE Conference on Mobile Computing and Networks (MobiCom)*, pages 203–214. ACM, 1998
- A. Ranganathan and R.H. Campbell. A middleware for context-aware agents in ubiquitous computing environments. In *Middleware*, pages 143–161, 2003
- Dieter Fox, Jeffrey Hightower, Lin Liao, Dirk Schulz, and Gaetano Borriello. Bayesian filtering for location estimation. *IEEE Pervasive Computing*, 2(3):24–33, 2003
- Jeffrey Hightower and Gaetano Borriello. Particle filters for location estimation in ubiquitous computing: A case study. In *UbiComp 2004: Ubiquitous Computing*, pages 88–106. Springer, 2004

7.7 Laboratory Exercises: Process and Store Location Data

The intention of this lab is to explore the processing of sensor data which reveals location information and to examine the use of R-Trees.

1. Identify a source of location data: GPS logs or way-marks; network-location traces, e.g. from Crawdad[7]; or even a distributed collection of person detection from a system as described in lab 8.7. You may need to perform some parsing and define data structures, or to use 3rd party tools and libraries for this.

[7] http://crawdad.cs.dartmouth.edu.

2. Process the tracks into an R-Tree implementation (start with the basic version). Again, you may wish to develop the code yourself from the algorithm or use a 3rd party source. Identify places stopped at and tracks / events.
3. Consider the efficiency of algorithms used in either log processing (as if real time) or R-Tree use: both CPU load and memory use matter in pervasive computing. Processing costs vary with algorithms and data structures used. In Listing 7.1 we present the linear cost algorithm for R-Tree insertion from [7], but you may wish to explore algorithms or R-Tree variants. The insertion cost and query costs over the resulting tree vary between algorithms, but also between data access patterns. Which solutions have which properties; do the properties hold across data sets of differing character?

7.8 Laboratory Exercises: Beacons and Location

The intention of this lab is to explore the use of beacons in forming location information. More sophisticated projects in this area might be easier to build using Place-Lab or other existing systems, but here we tackle the underlying issues of collecting data from beacons and interpreting it. We assume the use of the *BlueCove* Bluetooth stack and API here, as this is a free, Apache licensed, cross-platform, Java implementation. The principles are applicable to other implementations. This lab requires that Bluetooth devices are available: on a computer for running programs and some other nearby devices to act as beacons. There is no need for data exchange between these devices, but people with Bluetooth enabled around the working area should be informed so that they can check their security arrangements.

1. Download the BlueCove JAR[8] and open the API documentation[9].
2. Use the `RemoteDeviceDiscovery` example class from the API documentation as a basis, copy this to your development environment, add the JAR to the classpath and test. You should find a list of devices in the area.
3. Modify the code to provide (and possibly store) a scan at a set interval or on a prompt. Move about the area (maybe 4 m at a time initially) and form a map of which devices are visible where. Depending on the space and number of devices you may wish to extend the code to allow systematic mapping.
4. Review the data collected. Which areas in the space can be distinguished? Is a smaller distance between scans needed (to make more sense of data or to better reflect the space)?
5. Repeat the mapping and compare the two sets of data. Have any of the devices seen moved or vanished? Could the identity of these be predicted? For the remaining devices does the mapping show similar results?
6. As an extension to this it would be interesting to form a system which describes a statistical belief about location based on a database of survey results.

[8] http://code.google.com/p/bluecove.

[9] http://www.bluecove.org/apidocs.

7. Another extension would be to connect to specific types of beacons which describe their location. This could be achieved using GPS devices providing location data over Bluetooth in a space where GPS works; or by defining a Bluetooth service which gives some self-defined location description. The ability to infer the location of the client from the beacons' locations is interesting to study: certainly there may be separation between the beacon and client (many Bluetooth devices claim a 10 m range) but multiple readings may reduce this uncertainty.

7.9 Laboratory Exercises: Smart-Phone Location

If you are interested in the use of location data in practical scenarios, this lab integrates several concepts fro this chapter. Many smart-phones now integrate GPS receivers, QR code readers and the ability to run user code. The Android platform is a good starting point as the tool-set is free and quite simple to learn for those with the Java background assumed by this book and emulators are available. Working with a smart-phone platform is also good experience in working within the constraints of battery and store.

The specification here is quite general, and depending on your programme of study might be used as a larger scale project.

1. Access location data from your smart-phone, to show current location.
2. Pass your location data stream into one of the error modelling and filtering algorithms from Chap. 6. For some platforms, e.g. Android, the location data may come from GPS or cellular data and so have varying error patterns. Add to this a liveness detector, to identify when the data is old.
3. Report locations with accuracy data to a server and store or plot. Consider that the network may degrade at the same time as GPS signal degrades, e.g. in a tunnel and provide some local store in these situations.
4. Develop a set of behaviours to arise from specified locations. You may wish to match location to specific "points" and to more general areas. For the purposes of the lab these behaviours might simply be printing a message.
5. Consider how you would map from GPS coordinates to a higher level location scheme, such as place names. Consider whether place names might include user-specific places that map onto common conceptual places, e.g. home, work.
6. Consider how your behaviour specifications might work with the rest of the system, for instance: A hierarchical place name system might allow different levels of error to identify different places. In a space where there are no fine-grained specifications nearby is switching the GPS off to save energy a reasonable strategy?
7. Identify situations where the error is large or liveness poor and seek alternative input, such as QR code tags or user input. Consider that these may refer to a different coordinate scheme to the usual location model.
8. If you have other sensor data, such as an accelerometer, available consider whether this can inform your model of error and/or time-out of liveness.

References

1. Becker, C., Dürr, F.: On location models for ubiquitous computing. Pers. Ubiquitous Comput. **9**(1), 20–31 (2005)
2. Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B.: The R*-Tree: an efficient and robust access method for points and rectangles. In: Garcia-Molina, H., Jagadish, H.V. (eds.) SIGMOD Conference, pp. 322–331. ACM Press, New York (1990)
3. Beigl, M., Zimmer, T., Decker, C.: A location model for communicating and processing of context. Pers. Ubiquitous Comput. **6**(5/6), 341–357 (2002)
4. Bulusu, N., Heidemann, J., Estrin, D.: Gps-less low-cost outdoor localization for very small devices. IEEE Pers. Commun. **7**(5), 28–34 (2000) [see also IEEE Wireless Communications]
5. Egenhofer, M.J., Herring, J.R.: A mathematical framework for the definition of topological relationships. In: Brassel, K., Kishimoto, H. (eds.) 4th International Symposium on Spatial Data Handling, pp. 803–813. International Geographical Union, Zurich (1990)
6. Fox, D., Hightower, J., Liao, L., Schulz, D., Borriello, G.: Bayesian filtering for location estimation. IEEE Pervasive Comput. **2**(3), 24–33 (2003)
7. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: Yormark, B. (ed.) SIGMOD Conference, pp. 47–57. ACM Press, New York (1984)
8. Hightower, J., Borriello, G.: Location systems for ubiquitous computing. IEEE Comput. **34**(8), 57–66 (2001)
9. Hightower, J., Borriello, G.: Particle filters for location estimation in ubiquitous computing: a case study. In: UbiComp 2004: Ubiquitous Computing, pp. 88–106. Springer, Berlin (2004)
10. Hightower, J., Brumitt, B., Borriello, G.: The location stack: a layered model for location in ubiquitous computing. In: WMCSA, p. 22. IEEE Computer Society, Washington (2002)
11. Jiang, C., Steenkiste, P.: A hybrid location model with a computable location identifier for ubiquitous computing. In: Borriello, G., Holmquist, L.E. (eds.) Ubicomp. Lecture Notes in Computer Science, vol. 2498, pp. 246–263. Springer, Berlin (2002)
12. Kindberg, T., Pederson, T., Sukthankar, R.: Guest editors' introduction: labeling the world. IEEE Pervasive Comput. **9**(2), 8–10 (2010)
13. LaMarca, A., Chawathe, Y., Consolvo, S., Hightower, J., Smith, I.E., Scott, J., Sohn, T., Howard, J., Hughes, J., Potter, F., Tabert, J., Powledge, P., Borriello, G., Schilit, B.N.: Place lab: device positioning using radio beacons in the wild. In: Gellersen, H.-W., Want, R., Schmidt, A. (eds.) Pervasive. Lecture Notes in Computer Science, vol. 3468, pp. 116–133. Springer, Berlin (2005)
14. Leonhardt, U., Magee, J.: Towards a general location service for mobile environments. In: Third IEEE Workshop on Services in Distributed and Networked Environments, pp. 43–50. IEEE, New York (1996)
15. Leonhardt, U., Magee, J.: Multi-sensor location tracking. In: 4th ACM/IEEE Conference on Mobile Computing and Networks (MobiCom), pp. 203–214. ACM, New York (1998)
16. O'Neill, E., Kostakos, V., Kindberg, T., Fatah gen. Schieck, A., Penn, A., Fraser, D.S., Jones, T.: Instrumenting the city: developing methods for observing and understanding the digital cityscape. In: Dourish, P., Friday, A. (eds.) Ubicomp. Lecture Notes in Computer Science, vol. 4206, pp. 315–332. Springer, Berlin (2006)
17. Prakash, R., Baldoni, R.: Causality and the spatial-temporal ordering in mobile systems. Mob. Netw. Appl. **9**(5), 507–516 (2004)
18. Ranganathan, A., Campbell, R.H.: A middleware for context-aware agents in ubiquitous computing environments. In: Middleware, pp. 143–161 (2003)
19. Shekhar, S., Chawla, S.: Spatial Databases—A Tour. Pearson Education, Prentice Hall, Upper Saddle River (2003)
20. Survey, O.: The ellipsoid and the Transverse Mercator projection. Technical Report Geodetic information paper no. 1, version 2.2, Ordnance Survey (1998)
21. Want, R., Hopper, A., Falcao, V., Gibbons, J.: The active badge location system. ACM Trans. Inf. Syst. **10**(1), 91–102 (1992)