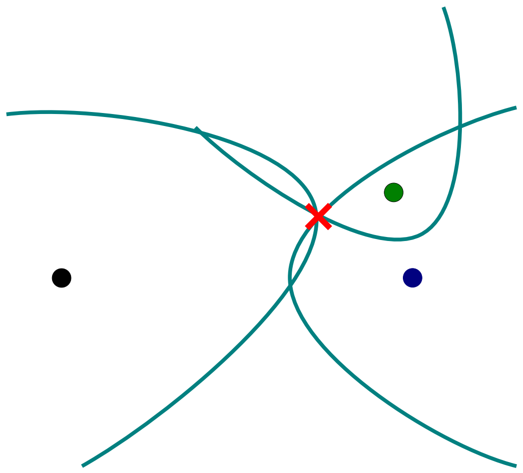
Additional Topics: Context- and Location-Aware Computing

Robert Harle

Easter Term 2010



Context and Location

Generally speaking, computers don't have a great reputation for easy interaction. Humans are extremely good at conveying concepts and ideas, but we struggle to achieve the same feat when communicating with computers.

If you think about human-human interaction, it is based not only on a rich language, but a shared understanding of how the world works (we leave a lot of things implicit). We also use gestures and facial expressions to increase our conversational bandwidth. We tend to lump together all this 'extra' stuff and call it *context*, and it's what computers typically lack. In fact, a traditional computer doesn't understand our language, has no understanding of the world (unless we explicitly model it) and can't determine the current context of the user, let alone understand it. We have to be excruciatingly explicit when communicating with computers, as well you know. Unfortunately this has meant that computers often become the focus of our tasks rather than the pure tools they should be.

Context-aware computing is an emerging research field that seeks to provide computers with more contextual clues, making interaction simpler or more natural. Simple context is pretty easy to derive. If I'm currently giving a powerpoint presentation (which is easily inferred by the OS), perhaps it's an inappropriate moment to alert me to my latest email, even if I have been "specially selected to receive a prize if you reply now". To infer more complex states requires us to add sensors to our computers, along with the software to make inferences from the sensor data. It is this latter approach that has captured the interest of researchers and industry alike.

It is instructive to think about what kind of contextual information might be of interest here. Dey identifies four types of context that are particularly important, known as the four ‘w’s: where, who, when and what. i.e. *location, identity, time* and *activity*. In a traditional computing scenario, these are relatively trivial to determine:

- Location: the user is located wherever the computer is (a static, known position).
- Identity: the user can presumably be identified as whoever logged in.
- Time: the time can be determined from the system clock and/or the network.
- Activity: the user is presumably typing (!).

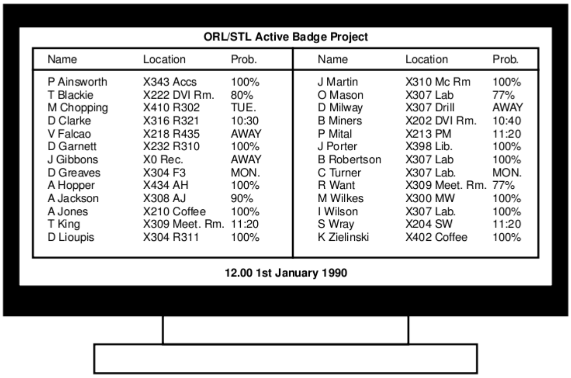
But this is a very restricted view of computers as physically anchored machines that we give exclusive attention to. Today, mobility it the hot area, and we are all carrying around one or two computers wherever we go. Mobility immediately makes location an uncertainty, and we might be ‘using’ a phone (i.e. have it switched on) without giving it our exclusive attention.

When you think about it, you’ll probably find that the biggest clue to your context is your location. If you’re in a lecture theatre it is highly unlikely that you are dancing, eating or getting dressed. There is, however, a high probability that you are writing and sitting (or, it would seem, sleeping). You’re probably not disturbable. We can infer a lot (both what you could and could not be doing) just from knowing *where* you are and possibly *who* you’re with. And how your location changes with time is also revealing: speeds tell us whether you’re in a car, walking or cycling for example.

For this reason, many of the demonstrated context-aware systems are based primarily on location information, and this looks set to be the NextBigThing^{TM}.

0.1 Example CA Systems

The Active Bat. Probably the first demonstration of a non-trivial, working context-aware system came from the Olivetti research labs here in Cambridge (headed by Professor Hopper). The *Active Badge* system used infra-red tags that employees wore in order to be tracked around the laboratories (more details on how it worked soon). The primary motivator came from the telephony system: the researchers were very mobile in



ORL/STL Active Badge Project

Name	Location	Prob.	Name	Location	Prob.
P Ainsworth	X343 Accs	100%	J Martin	X310 Mc Rm	100%
T Blackie	X222 DVI Rm.	80%	O Mason	X307 Lab	77%
M Chopping	X410 R302	TUE.	D Milway	X307 Drill	AWAY
D Clarke	X316 R321	10:30	B Miners	X202 DVI Rm.	10:40
V Falcao	X218 R435	AWAY	P Mital	X213 PM	11:20
D Garnett	X232 R310	100%	J Porter	X398 Lib.	100%
J Gibbons	X0 Rec.	AWAY	B Robertson	X307 Lab	100%
D Greaves	X304 F3	MON.	C Turner	X307 Lab.	MON.
A Hopper	X434 AH	100%	R Want	X309 Meet. Rm.	77%
A Jackson	X308 AJ	90%	M Wilkes	X300 MW	100%
A Jones	X210 Coffee	100%	I Wilson	X307 Lab.	100%
T King	X309 Meet. Rm.	11:20	S Wray	X204 SW	11:20
D Lioupis	X304 R311	100%	K Zielinski	X402 Coffee	100%

12.00 1st January 1990

Figure 1: An Active badge tracking application

their working, constantly moving between offices, meetings and hardware labs and they found that they were missing many phonecalls because they were away from their designated office. The Active Badges provided the phone routing system with the necessary context to be able to automatically route a call to the phone nearest to the user rather than to their desk.

Users quickly took to the idea and applications were developed to allow people to see where their colleagues were at any time (Figure 1). Attempts were also made to support 'desktop teleporting': your computer desktop was automatically transmitted to the nearest computer workstation to you using the VNC protocol[1].

From the AT&T Archives: [2]

> "Over 1500 badges and 2000 sensors are deployed throughout a number of European universities including the University of Kent, Imperial College, London, Lancaster University, and the University of Twente, Netherlands. In the USA, Xerox PARC, DEC research laboratories, Bellcore and MIT Media Lab have all received Active Badge systems.

[1]If you're not familiar with VNC, think of it as a application that recreates your computer desktop at a remote site by constantly sending images of it over the network, similar to Windows remote desktop (RDP)

[2]http://www.cl.cam.ac.uk/research/dtg/attarchive/ab.html

The largest single system is at Cambridge University Computer Laboratory , where over 200 badges and 300 sensors are in daily use. Information about the location of individuals is also exchanged between these sites where appropriate."

Tour Guides. Somehow, the canonical demonstrator of context-aware systems became the automatic tour guide. Typically, users carry a mobile device as they travel through an unknown area (perhaps a city or a museum). The intention is that the device adapts to the user's current situation; perhaps telling them where they are or what landmarks are in front of them. These devices also infer context from the behaviour of their current user. So, for example, if the current user avoids going into the suggested art museums, the guide might stop suggesting such museums or tailor the information it gives to concentrate on aspects other than art. Again, we see that location information is key.

Reminders. Many researchers are attracted by the idea of context-aware reminders. In some ways, an alarm clock is a very simplistic context-aware reminder (since it hopefully tracks the time). More common examples usually include reminding users based on their location. So, for example, a system might remind you that you have a meeting in ten minutes if it infers that you are getting into your car to leave, or it might remind you that you need to pick up some milk as you drive past the supermarket. This sort of thing can be rather tough to implement since too many reminders are rather annoying. In the meeting example just given, you might simply be going to retrieve something from your car in preparation for the meeting, and the reminder would be rather annoying..! But again, location is giving us the context.

Environmental Controls. Probably the most pervasive deployment of context-awareness is the use of infra-red presence detectors to control the environment based on the users within it. Usually these are used to switch lights on and off automatically, although they do get used for other tasks too (public toilets, for example, use them to estimate the number of people using the facilities to inform the cleaners). In fact, these systems emphasise a useful point. We have all been sat in rooms that use them when the lights go out because the system incorrectly thinks we've gone. Because there's no conventional interaction system (the switch has usually been replaced), this is a source of much frustration. When designing these systems we need to take care that the context is reliable or, if not, that there is a straightforward override.

Generally, equipping buildings with sensors to optimise the infrastructure is becoming very popular. And, yet again, user location is the major source of context.

Given how important location is, much of the research to date has concentrated on it. For the remainder of this lecture and next, we'll be looking at *how* users (or, more accurately, the devices they carry) can be located

Location Techniques

0.2 GPS and Accuracy

Almost everyone is familiar with GPS these days. It's become such an important system that so many of our computer systems rely on (usually for providing accurate time, but also for location) that you rely ought to have an understanding of how it works. Later in the course Dr Alan Jones will give an entire lecture on GPS. For now it is enough to know that GPS provides us with ubiquitous location *outdoors only* to tens of metres.

However, we spend much of our lives indoors, out of the way of GPS signals. But even if they did penetrate indoors, the results probably wouldn't be much use. For most GPS applications, the world is considered to be a 2D map of immense size. The landmarks of interest on that map are usually very well separated by tens, hundreds, even thousands of metres. Thus a GPS location with a 15 m accuracy gives us all we need.

Indoors, however, is quite different. Buildings have floors so 2D mapping is out. A GPS fix accurate to 15 m (in 3D) isn't so useful here: you can't determine the floor the user is on, never mind the office they are in!

In my experience, indoor location is useful on two different scales:

Room level. Knowledge of the room we are in (and probably those others we are with) says a lot about our context. Many of the devices we use can be considered on a room scale. For example, we can imagine computers that unlock when we are in the room, phonecalls that route automatically and lighting that responds to our presence.

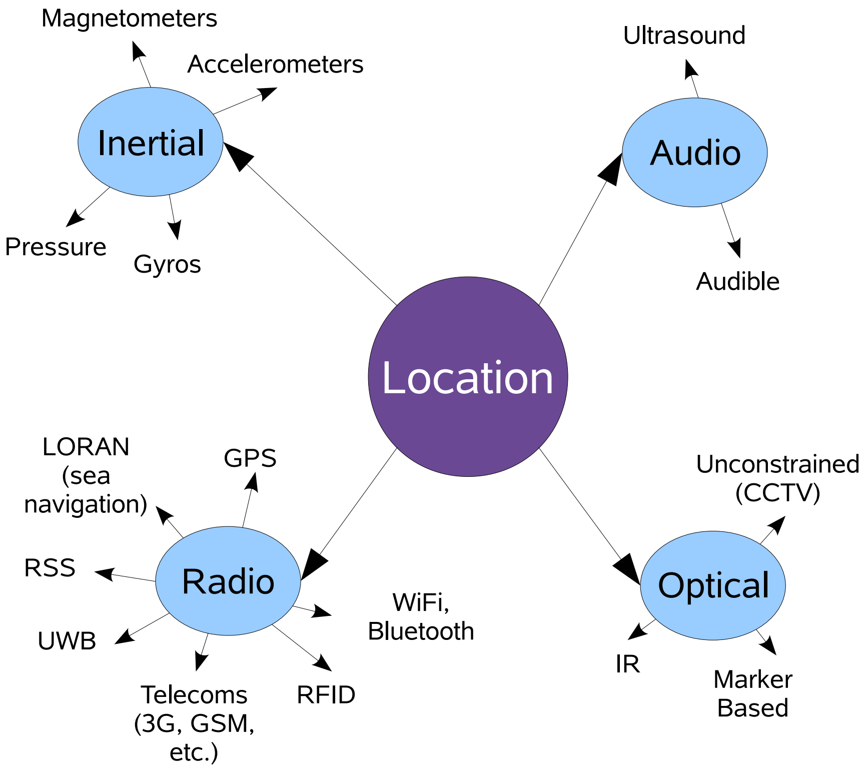
Sub-metre level. We can also imagine more precise location providing room devices with better context. Perhaps a computer should not unlock itself unless I am directly in front of it (not just in the corner of the room), or the phone selection algorithm may wish to distinguish between multiple phones in the room for call routing. Typically, our devices or spaces of influence are separated by a metre or so and thus we need sub-metre accuracies to exploit this context.

0.3 What do we Measure?

Very rarely can we just measure ‘location’ directly (the only example that comes to mind is a tape measure and that isn’t great for tracking your car...). Instead we use measurable quantities to derive location estimates.

So what exactly do you measure? Over the years we have seen location systems that use a variety of different physical phenomena to infer location. The diagram below provides a feel for the different media available to derive location. They all have their advantages and disadvantages and many can be used in different ways.

Our approach herein will be to look at specific classifications of location system, discussing the underlying principles and giving some concrete examples.


Magnetometers
Accelerometers
Inertial
Pressure
Gyros
Ultrasound
Audio
Audible
Location
LORAN (sea navigation)
GPS
RSS
Radio
UWB
Telecoms (3G, GSM, etc.)
RFID
WiFi, Bluetooth
Unconstrained (CCTV)
Optical
IR
Marker Based

Proximity-Based Systems

0.4 Principles

The accuracy of the location information we need depends heavily on the scale of our application(s) and on the mobility of the subject. Some applications only demand coarse accuracy. For example, if I'm tracking a product through the delivery chain, there is a small set of places the object might be (distribution centre, packing room, delivery van, etc.). It's unlikely to be important to locate it more accurately than to a specific building.

When we require coarse localisation to one of a set of well-defined locations, proximity-based tracking works nicely. We seek not to pinpoint the device location but simply to say "it's near here", where 'near' is within some predefined range. RFID tags are a good example—if you can see tag T from Reader R then you can be confident that T is in the same building (and probably the same room) as R. Proximity systems can be based on any medium that has a limitable range—infra-red light and radio signals are common choices.

0.5 Case Studies

0.5.1 Active Badge (1989–1992)

The Active Badge system used small, powered (hence 'active') tags that were worn as badges (Figure 2). Each badge had a unique identifier and would periodically (0.1 Hz) transmit it over an infra-red (IR) channel.

Networked IR receivers were put up, roughly one per office. The great thing

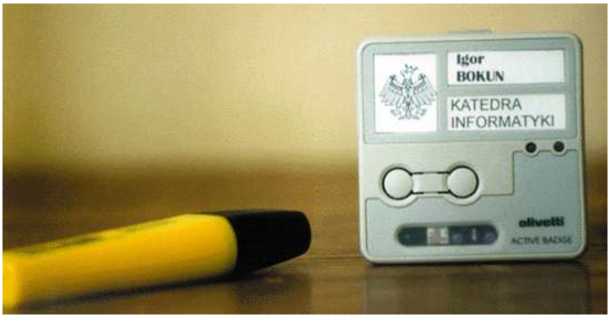


Figure 2: An Active Badge

about IR is that it will bounce all over the room before it dissipates (you know this from TV remotes) and doesn't penetrate the walls. This means we get natural room containment and so we can reliably associate users with rooms. It does mean we need at least one receiver per room, but we are not particularly sensitive to where that sensor is sited.

As discussed, the system was extensively deployed and was a great success. Occasionally there were problems in strong sunlight (which contains IR that would overpower the sensors) but generally the system worked well and was popular.

You might reasonably ask why we are not all carrying around Active Badges or their descendents these days. On reflection, I think it was ahead of its time and suffered for it. When it was demonstrated, the technology was only just capable of supporting the system. A redesign now would make the tags smaller, cheaper and with much longer battery lifetimes. Remember that it was developed in a world where global networking was still around the corner and location-awareness had never really been considered. Ironically, had it been invented today, I suspect it would be much more prevalent.

0.5.2 Radio Frequency Identification (RFID) Systems

Passive (i.e. unpowered) RFID tags have potential for tracking since they are very cheap and don't need batteries charged or changed. They can make a proximity based location system if you deploy a lot of RFID readers at known locations and have users carry the tags.

Unfortunately, radio waves penetrate most walls which means that we can't use them to localise users to specific rooms, only specific regions of the building. This is often a problem since most applications need to know for certain which room you are in: a set of possible rooms isn't usually good enough.

0.5.3 Bluetooth and WiFi

Bluetooth (and increasingly WiFi) is ubiquitous on modern mobile platforms, which makes it rather attractive for locating someone. The simplest approach is to use a proximity system: if a base station can see you, then you must be within range of it. Software like BlueProximity will do this for you.

Now, Bluetooth comes in three flavours, which have different transmit powers and hence different ranges. Nominally we have:

Class	Max Power	Range
1	100mW	100 m
2	2.5 mW	10 m
3	1 mW	1 m

It depends on the scale of your problem as to which you want to use. If it's indoors, locating someone to a radius of 100 m probably doesn't even get you the building they're in! A radius of 10 m is better, but it's really only going to get you 'portion of a building' accuracy (like RFID, it will be hard to pinpoint people to specific rooms because the radio penetrates the walls). You'll also be needing lots of Bluetooth hosts to cover an entire building.

In reality, creating a Bluetooth tracking system isn't trivial. The simplest method that works for any device is to leave the mobile device discoverable and have every host constantly scan for in-range devices. This is generally bad because:

- Bluetooth discovery sucks. To discover all the devices in range, you must let each discovery query run for 10.24 s (this is all to do with power saving at the mobile end). It means you get a pretty awful update rate for tracking.
- Discoverability is considered a security risk.

- Most modern phones/devices won't even *allow* you to leave discoverability on indefinitely because of the previous point!

Nonetheless, some have had success. The website http://www.bluetoothtracking.org leaves a scanning station next to a highway, and reports that it sees 3,200+ handsets an hour at peak times! This isn't so much tracking as an instantaneous measure of position, but it tells you something about what's out there...

WiFi is a little better on these counts, since security is stronger and discovery is comparably fast. But WiFi is very power hungry compared to Bluetooth (as any iPhone user will tell you). So you want to turn off WiFi as much as possible, not have it permanently on!

As it turns out, WiFi generally has a pretty big range, and we often get access points with overlapping coverage. In this case, we can use *radio fingerprinting* to locate with more accuracy—see later!

0.5.4 Serving Cell Phone Location

The mobile phone network is designed such that your phone talks to the strongest base station that it can hear at any given time—this base station is known as the *serving cell* of your handset.

For mobile telephony networks, the strongest station is almost always the nearest. Therefore, the network operator can localise any phone to within the range of its serving cell—proximity-based location!

How accurate it is depends heavily on the serving cell and its location. In rural areas, cells are sparse and so their coverage is very large (many km). In built up areas, lower power (and hence smaller) cells are often used in a more dense distribution, giving better localisation. We are still talking many tens or hundreds of metres, though.

AoA Systems

0.6 Principles

Most people think of the word 'triangulation' when they are asked to compute a position. Thing is, not many people seem to understand what it means and, like chinese whispers, it has ended up with a lax and unclear definition.

We're going to take a reasonably strict definition. Triangulation applies to so-called 'Angle of Arrival' (AoA) systems where we can somehow measure the angle at which a signal hits us from a source we are trying to locate.

The principle is simple. Take two measurement stations at **A** and **B** and an object to be located at **P**. Assume the object is transmitting in some way (usually but not always radio). The two stations measure the incident signal angle and form a triangle in 2D space based on the two bearings and **A** and **B**. The third triangle vertex will be at **P** (see Figure 3).

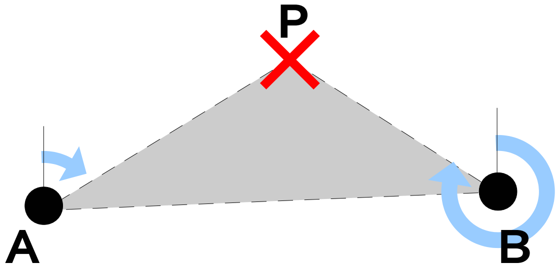


Figure 3: Triangulation basics.

0.6.1 How can you measure the AoA?

Typically we would use an antenna array and measure the phase difference between signals received at different elements.

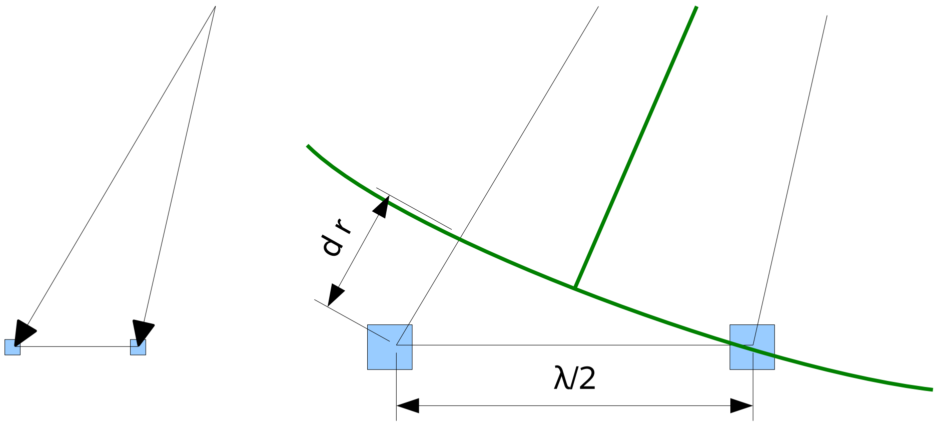


Figure 4: A two-element antenna array

For example, consider a two-element array with element separation of one half-wavelength (Figure 4). As the source bearing changes, the signal hits each element at slightly different times due to different path lengths (say, a difference of dr). Thus the signals at the two elements may have a different phase. If they have the same phase, the source must be at the top of an equilateral triangle. $\mid dr \mid$ has a maximum of half the wavelength (making the signals exactly out of phase) which occurs only when the array is parallel to the bearing to the source. All measured phases in between imply a different transmitter bearing.

0.6.2 Siting the Stations

Every measurement has error associated with it, so it's interesting to think about how sensitive this approach is to errors in the bearings. This is all about the geometry.

Consider the vectors from the base stations to $\mathbf{P}$. If these vectors are near parallel (similar bearings), the triangle must be very tall and thin. A small error in bearing gives a big change in the estimate of $\mathbf{P}$. Conversely, if the vectors are near perpendicular, a small bearing error doesn't have so great an effect on the estimated $\mathbf{P}$. Thus the geometry of our stations relative to our source is very important.

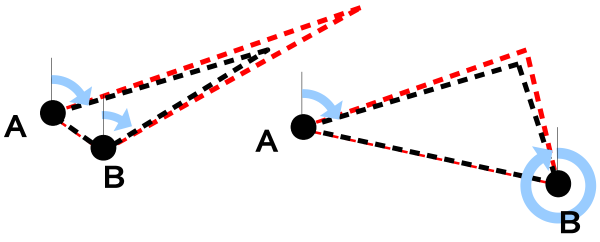


Figure 5: Large errors in position can result from small errors in measured bearings.

0.6.3 Multipath

In the real world there's another problem to worry about: *multipath*. This is the term used to indicate that a signal propagates from source to receiver via multiple paths, usually due to reflections (Figure 6).

In an AoA system, this is a major problem if the direct (*line-of-sight*) path doesn't get through, but a bounced signal does. Now the bearing is all wrong and we get garbage in our location calculation (Figure 7).

How do we address this? We put redundancy into our system. In 2D we only

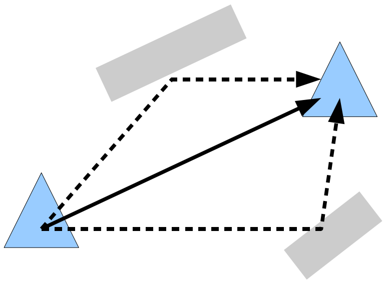


Figure 6: Multipathed signals caused by reflections

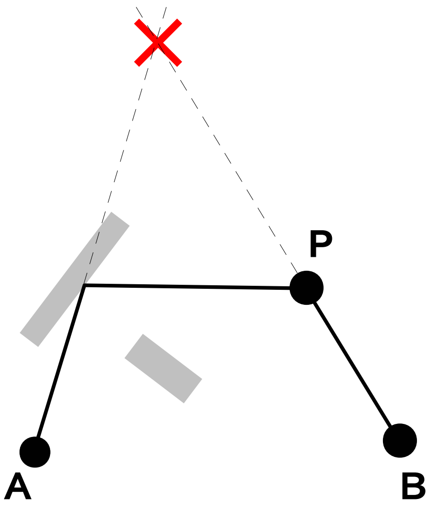


Figure 7: Why multipath breaks AoA estimates

need two bearings in principle, but we actually use as many as possible. We are then applying a *multiangulation* approach whereby we process an over-determined system, looking for consistency in the data. E.g. six bearings may be taken, five of which agree and one which does not; we can throw out the latter as multipath.

If we can, we distribute the measurement positions *around* the source to minimise the effects of bearing errors.

0.7 Case Studies

0.7.1 Pirate Radio and Enemy Transmitters

AoA systems are usually used whenever you want to locate the source of a transmission over which you have no control and you don't have the luxury of having a permanent set of listening stations surrounding the transmitter (if you do you can use TDOA—see later).

Say you are trying to localise a radio transmission in the desert from your helicopter (you *do* have a helicopter, right?). You take a bearing to the signal from wherever you are (GPS tells you that info). Then you fly perpendicular to that bearing for a bit and take a new bearing at a new position. You should now be able to locate the source (modulo errors) using AoA.

You can use a similar approach to locate a pirate radio transmitter. First you tune to the station and get a bearing to the source. Then you move around a bit and repeat. You use the two measurements you have to very roughly estimate the transmitter location and then move to another location that gives you the best geometry to pinpoint it more accurately with AoA.

ToA or ToF systems

0.8 Principles

Our next class of location system is a *Time of Arrival* (ToA) or equivalently a *Time of Flight* (ToF) system. The idea is that we somehow measure how *long* it takes for a signal at the source to reach a set of receiver stations at known locations.

Times don't help us much directly, so we convert to distances on the assumption that we know the speed at which the signal travelled. Again we are looking to form a triangle to get a position, but instead of having the angles we now have the triangle side lengths. Computing the source position from this information is *trilateration*.

Imagine our stations are at $\mathbf{A}$ and $\mathbf{B}$, and a signal (propagating at speed c) arrives at times t_A and t_B, respectively. Then we know:

$$|\,\mathbf{P} - \mathbf{A}\,| = ct_A \tag{1}$$

$$|\,\mathbf{P} - \mathbf{B}\,| = ct_B \tag{2}$$

You can think of this as intersecting circles of set radii centred on the station locations (the above equations each describe a circle in 2D). The problem is that this almost always gives ambiguity in position (Figure 8). To solve this, we must ensure that we have a minimum of three measurement stations (Figure 9).

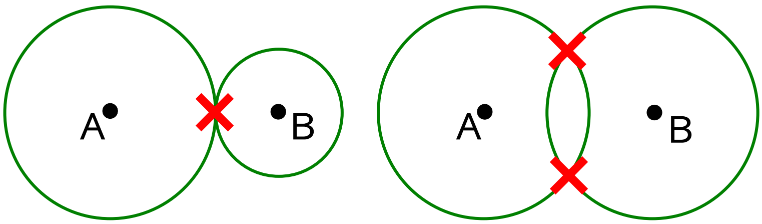


Figure 8: Ambiguity in trilateration.

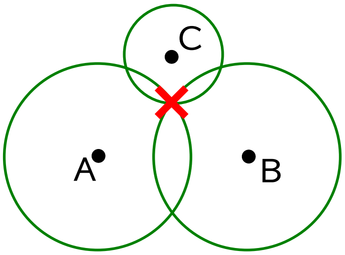


Figure 9: Three measurements removes ambiguity.

0.8.1 Noise and Geometry

What about error in the timings? As with angulation, the magnitude of the effect this has is dependent on the geometry of the receivers. The ideal geometry for 2D has three receivers, each at a vertex of an equilateral triangle that contains $\mathbf{P}$.

Problems arise when the vectors between the receivers and the source are close to parallel. If this happens, a small change in circle radius can have a big effect on the intersection points (Figure 10)

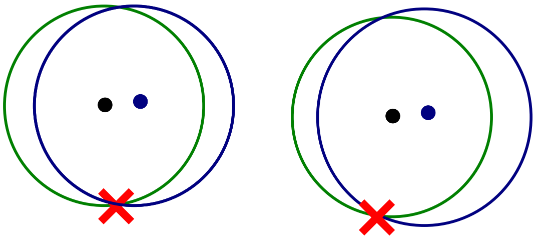


Figure 10: Bad ToA Geometry.

0.8.2 Synchronisation and Timing

ToF systems are problematic because you want multiple receivers to time how long a signal takes to get to them. This comes with at least two issues:

Station Sync. The receiving stations have to start their virtual stopwatches at exactly the same moment as the source emits. This means that the source has to be under control. Even then, synchronising clocks across multiple sites with sufficient accuracy is *not* easy.

Timing Accuracy. Assume we are dealing with radio waves propagating at the speed of light. If we want our timings to be within just 10 m of the correct value, we need to be able to time to accuracies of 10/300,000,000 = 33.333 *nano*seconds. That's expensive kit...

Additionally, even when you have an accurate clock, when do you stop it? Realistically we don't get a perfect pulse to trigger the action, and the channel may have noise that could trigger it wrongly if we're just doing some sort of threshold.

Instead, we can rely on the same signal being received at each receiver, just at different times. We sample the received signals at high rate and then *cross-correlate* the results to figure out the timing difference:

$$[f \star g]_i = \int_{j=-\infty}^{\infty} f_j g_{i+j} \tag{3}$$

where f and g represent the sampled signals at two different receivers.

0.8.3 Multipath

Just like with angulation, multipathed signals cause headaches. They cause measured ToF values to be larger than they should be (never smaller) and thus cause distances to be overestimated.

The solution is the same as with angulation, however. *Multilateration* uses ToF readings from more than three spatially distinct receivers to estimate the position and to throw out outliers (multipathed signals).

0.9 Case Studies

0.9.1 The Bat System

The Bat system (originally “Active Bat System”) came out of a CUCL PhD by Andy Ward that was developed by AT&T Research. It is a ToA system that times ultrasonic pulses from small, wearable devices (“Bats”) to a set of receivers in the ceiling. A radio system (433 MHz) polls a Bat, telling it to ‘squeak’ ultrasonically. In operation, the system starts a virtual stopwatch, polls a Bat and waits to hear the times that different ceiling receivers heard the squeak.

The choice of ultrasound is important for a number of reasons:

Easy Synchronisation. The great thing about ultrasound is that it moves approximately a million times slower than radio waves do (330 m/s vs 300,000,000 m/s). If the system starts a clock and polls a Bat 30 m away, the Bat receives the instruction after 10^{-7}s. If we ignore this transmit time altogether, we only introduce a lateration error of 0.000033m. In essence, we can get away with treating the radio propagation as instantaneous.

Easy Containment. One of the nice properties of IR for the Active Badge was that it is naturally contained within bounded spaces (rooms). Ultrasound has this property too, so we can pinpoint the correct room without even having enough data to trilaterate!

But beware: whereas IR bouncing all over the room was good news for the Active Badge, it’s potentially bad news for the Bat system since we care not only that the signal gets to a receiver, but that it travels directly and doesn’t end up multipathed. This was some of the thinking behind putting the receivers into the ceiling—if you are wearing a bat at chest height, you will usually have a direct path to the ceiling.

Performance

- The Bat system achieves **3cm accuracy** *in 3D space* 95% of the time!

- The position update rate is variable, with a maximum of around 15 Hz. This is a nominal value chosen to ensure that each ultrasonic pulse has fully dissipated before the next is sent.

Deployment

The Bat system was deployed across three floors of the old AT&T Research building near Engineering. It was also deployed in a single room in engineering, and subsequently along the entire length of the SN corridor in the WGB (i.e. the DTG area). Today, it still runs in the WGB and is used for location research (usually as a ground truth).

Issues

The Bat system is arguably the most accurate large-scale person-tracking in existence, but it isn't perfect. One of the problems with a system that can potentially achieve cm-precision is that *the accuracy to which you can locate your receivers becomes a limiting factor*! We want to get the receiver locations measured with an accuracy that is an order of magnitude smaller than the expected location accuracy. That means location to a few millimetres across hundreds of square metres. Good luck with that...

In the current deployment we used laser surveying stations (the type that architects use) and went to great effort. Realistically we probably measured to within 15 mm. Over time, however, receivers are bound to move or be knocked and that accuracy has doubtless faded.

The next issue concerns the number of receivers. Ultrasonic containment is nice on one hand, but means that wherever you face in a room, there must be at least three receivers in the ceiling to get a position fix. That means you need a lot of receivers (all accurately positioned!). The 550 m^2 deployment in the WGB (that's 23 rooms or corridor areas) uses a whopping 409 receivers, all carefully surveyed..!

0.9.2 MIT's Cricket

MIT created the Cricket indoor localisation system that also uses ultrasonic pulses. They didn't like the centralised nature of the Bat system, nor what they perceived as an inherent lack of privacy (the system knows where you are and you have to ask it).

Cricket (in its original form) uses beacons installed around an office. These beacons usually aren't networked and they periodically send out a radio pulse and (simultaneously) an ultrasonic pulse. Cricket devices measure the time difference between receiving a radio pulse and its corresponding ultrasonic pulse, deriving a ToF for the ultrasound on the assumption radio is instantaneous.

You can use this system in a number of ways. You can have a sparse set of beacons in approximate locations and use it as a proximity system, or a dense set of beacons at carefully measured locations and use it as an inverted Bat system. If you do the latter, you can potentially achieve similar accuracies, but you do end up with the same deployment issues that the Bat system has.

Inverting the Bat system so that the device locates itself makes the system more scalable (no centralised multilateration calculations) but puts a serious load on the mobile devices so they eat up their batteries faster and need more grunt.

The privacy argument is interesting (no-one has your position unless you choose to give it to them), but ultimately limiting. If you know your location, that's great for standard mapping (where am I? How do I get to..?) but if you actually want to have the ubiquitous computing/location-aware benefits, you end up having to continually report your location to a central body anyway...

TDOA Systems

0.10 Principles

Synchronisation is a big issue in ToA systems—to time how long it takes for some signal to propagate between two points means that we have to have a clock at each site and those clocks must be synchronised. In the real world, we're pretty good at synchronising two systems when there's a reliable piece of wire between them (think NTP and better).

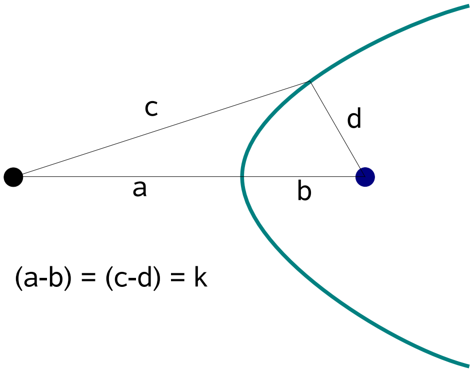
For a location system, we have a problem. The locatable device needs to be mobile (or you don't need a location system!) and mobile devices won't have the serious hardware you need to synchronise two systems together to nanosecond precision over radio. So, pure radio systems can't realistically synchronise the mobile node with the receivers, and ToA doesn't work.

Instead we can synchronise our receivers together (usually with bits of wire) and use a *Time Difference of Arrival* (TDOA) system to handle the fact that we can't know precisely when the mobile node transmits.

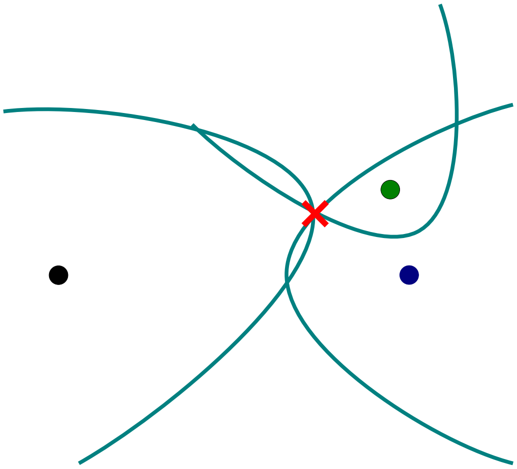
The method is best illustrated by example. Take our stations $\mathbf{A}$ and $\mathbf{B}$ and assume they log the same signal at times t_A and t_B (note these times are in the same frame of reference, such as GMT, but are *not* the ToF values for the signal because we don't know when the signal was sent). If we assume $t_B > t_A$ then we can state that station B is $c(t_B - t_A)$ further away from P than A. i.e.

$$|\, \mathbf{P} - \mathbf{B} \,| - |\, \mathbf{P} - \mathbf{A} \,| = c(t_B - t_A) \tag{4}$$

This is actually the definition of a hyperbola with centre at the midpoint between the two stations. i.e. From a pair of stations we can restrict **P** to lie on a hyperbola in 2D space despite not knowing when the transmission began.



Now all we do is look at multiple pairs of receivers to derive multiple hyperbolae and look for the intersection in the same way as we did with circles in a ToA system.



Note that we need three pairings to get three hyperbolae for an unambiguous 2D fix, but that we can get these from just three base stations since the pairings need not be completely independent.

Additionally, there is no requirement for the measurements for all pairings to be derived from the same signal. We might get the A-B pairing one second, and the B-C the next. This is fine, so long as the object isn't going to move significantly in the time it takes to collect all of the pairings you want.

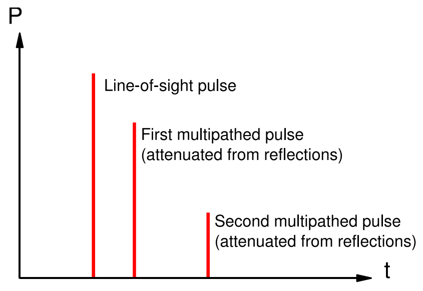
0.11 Case Studies

0.11.1 Ubisense

The folks that developed the Bat System and its associated software went on to found Ubisense, which makes a location system that is essentially the Bat system, but with the ultrasound replaced with *ultra-wideband radio* (UWB[3]). This is a signal that has a very wide range of frequencies in it. By ‘very wide’ we are talking the entire range from 1 GHz to 10 GHz.

UWB is seen as the next big thing because it distributes its power across a very big range, meaning there’s only a little bit of power at each frequency. So little, in fact, that standard narrowband equipment is meant to treat it as noise and ignore it. The result is that it can provide very high bandwidths and co-exist with current radio systems. It is currently proposed as a way to replace the spaghetti behind every desktop computer (even the VGA cable...).

But all of that is irrelevant for location. When you want to make an indoor radio system, you usually need two things; accurate clocks and very, very short pulse durations. The reason you need sharp pulses is the extensive multipath indoors, and many of the path lengths differ by very little. This means that the direct signal is followed closely in time by bounced paths. So you might be expecting to see something like this at a receiver if you sent out a pulse:



[3]http://en.wikipedia.org/wiki/Ultra-wideband

Now, think back to your mathematical methods course (or DSP if you did it), in particular Fourier Transforms. If you Fourier Transform a signal in the time domain then you get the frequency domain representation (i.e. the frequency spectrum of that signal):

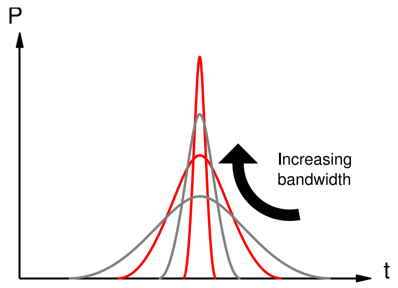
$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt. \tag{5}$$

If we consider stretching or compressing the time domain by some factor A we find

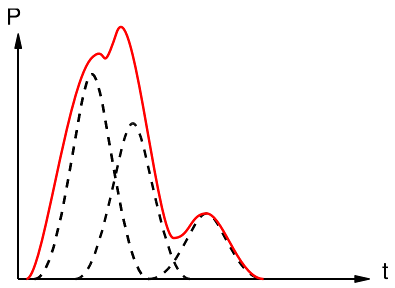
$$f(At) \leftrightarrow \frac{1}{\mid A \mid} F(\omega/A) \tag{6}$$

i.e. if you want to squash the time domain by A you need to *stretch* the bandwidth (range of frequencies) by factor A. Now, a perfect, infinitesimal pulse requires *every* frequency (zero time duration gives infinite frequency bandwidth). In the real world, our hardware can only produce a signal with a limited bandwidth, and thus a minimum pulse duration.

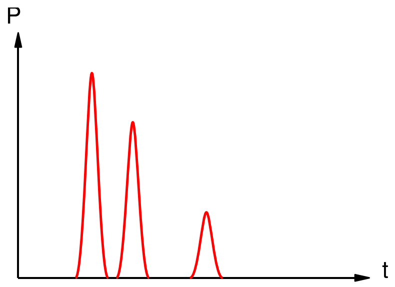
So, if you have a small signal bandwidth (non-UWB signals), you can't create very sharp pulses. The more bandwidth you add the sharper these pulses get (UWB can manage nanosecond pulse widths):



Now, if you let the signal bounce around as it would in a multipathed environment, you get a complete mess if the pulse width exceeds the timing between successive paths:



For UWB, we only get overlap if the path differences are of the order of 0.5 m or less. Typical indoor environments produce path differences of order of a metre, so UWB usually allows us to identify the first pulse accurately. ‘Normal’ radio systems can’t get the pulses thin enough. A UWB pulse in a multipathed environment might result in a receiver hearing something more like:



System Design

The Ubisense system design is similar to the Bat system - a narrowband radio channel is used to poll active tags that then emit UWB pulses that are received by an array of receivers. The Bat system uses ultrasound and radio so that it can ignore the synchronisation problem. Sadly, UWB moves at the same speed as the polling signal so the Ubisense system must use TDOA with pairs of receivers.

Actually, the system is a bit smarter than that because the receivers use a UWB antenna array. This means that it can *also* apply AoA analysis. In fact, if you assume the height of the tag you can even estimate 3D location from a single receiver (this is just used as a backup in case only one is visible—you never deploy it such that it always has to work in this mode!).

Deployment

UWB radio isn't contained by rooms. Whilst this means we could get the wrong room when positioning, it also means we don't need anything like as many receivers as we did with the Bat system. You can cover a few rooms with four well-placed receivers. Commercially-speaking, this is a winner.

Performance

The synchronisation task is more complex with a radio system (the ToF pulse moves at the same speed at the polling pulse) and radio is just hard to work with. The Ubisense system (when carefully configured) tracks with accuracies around 20 cm in 3D (worse in single sensor mode). It can also achieve update rates in the hundreds of Hz.

0.12 Mobile Phone Tracking

Hollywood would have you believe that you can be tracked so accurately through your mobile phone that they'd know if you tripped. Fortunately (or perhaps unfortunately?) they can't. But it's interesting to know what *can* be done.

Firstly, you need to understand some terminology: the network operator has a series of *Base Transmitting Stations (BTSs)*; your phone is a *Mobile Station (MS)*. GSM communication uses Time Division Multiple Access (TDMA) i.e. there are set time slots during which only one thing talks to the BTS.

How does an MS synchronise with a BTS so that it talks at the right moment? Each BTS regularly sends out a *synchronisation burst* which the MS can use to 'lock on' its clock. This means the BTS and MS are reasonably well synchronised (actually the BTS buffers the timeslots to allow for sync errors), but note that any two BTSs are *not* synchronised at all.

0.12.1 U-TDOA Phone Location

Uplink Time Difference of Arrival (U-TDOA) has been adopted by all the major US phone providers in response to the E-911 government mandate there (this is a law that requires a mobile phone to be locatable to various accuracies when the emergency 911 number is called).

It is basically standard TDOA on a mobile phone signal, except that we need to augment the BTSs with some kit to sync them up. This kit is deployed by the operator and is called a *Location Measurement Unit (LMUs)*. To save money, operators usually deploy LMUs at only a subset of their BTSs (the more the better as far as accuracy goes).

Each LMU monitors the signals received by the attached BTS and uses GPS to timestamp them in a global time frame. To position, the primary LMU for a given MS (usually just the closest) collects the receive times from LMUs using the data network, computes the time differences and thus a location using TDOA.

0.12.2 E-OTD Phone Location

A Cambridge company (Cambridge Positioning Systems, now part of Cambridge Silicon Radio) developed a technique known as Enhanced-Observed Time Difference (E-OTD) which is really a kind of constrained TDOA. The main change is an inversion of the system so that the BTSs transmit and the MS receives the signals used for positioning. The main difficulty now is that the BTSs do not transmit simultaneously.

The first piece of information we need is the set of times that the BTSs transmitted their bursts. An LMU in the network hears a set of bursts and uses the position information of itself and the transmitting BTSs (good old GPS again) to figure out when the relevant BTSs transmitted. So, BTSs A, B and C might transmit at absolute times t_a, t_b and t_c.

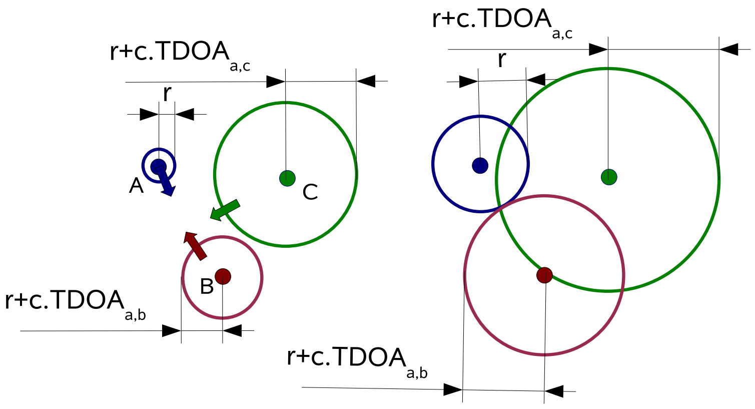
Meanwhile, the MS also hears the signals and, not having a global time reference, measures the differences in the reception times relative to a reference BTS with an LMU attached (say, A): $\Delta t_{a,b}$, $\Delta t_{a,c}$.

We can now compute the TDOA values:

$$TDOA_{a,b} = \Delta t_{a,b} - (t_b - t_a) \quad (7)$$
$$TDOA_{a,c} = \Delta t_{a,c} - (t_c - t_a) \quad (8)$$

At this point we could compute $TDOA_{b,c}$ and apply the usual TDOA calculation. This would be fine. However, when we have every BTS seen by a single reference BTS (as is always the case with E-OTD: the LMU BTS has to see all the others) we can apply a different analysis which *may* be easier to understand/easier to implement:

- The only real unknown here is the time it takes for a signal to get from A to the MS.
- Let's just set this arbitrarily to r. As with TOA, we can draw a circle around S with radius r
- Now we can draw circles around B and C with radii $r + c.TDOA_{a,b}$ and $(r + c.TDOA_{a,c})$, respectively.
- If we vary r until the three circles meet, we have our location!



Comparison

- In principle, E-OTD needs fewer LMUs deployed which means lower deployment costs. U-TDOA is very expensive to deploy.

- U-TDOA can deploy more sensitive receiving equipment on its LMUs and thus more BTSs will hear the phone than vice-versa. In principle an E-OTD LMU could transmit at greater power, but of course this is not allowed by regulations.
- U-TDOA typically achieves sub-80 m accuracy and can use 40+ BTSs per position (greater redundancy gives greater accuracy). E-OTD typically uses around 8 BTSs per position and achieves accuracies closer to 150 m.
- E-OTD only works on modified handsets, U-TDOA works on all.
- E-OTD accuracy is dictated by the handset capabilities (clock, processing, etc.). U-TDOA can use more powerful, bulky equipment.
- E-OTD requires the active participation of the handset so has a natural privacy-preserving mechanism. U-TDOA can be performed without the MS owner knowing.

Many US operators adopted E-OTD a few years back, but then decided that it couldn't reach the accuracies that it had to reach for the FCC e911 mandate. The result is that most US operators have now coughed up and use U-TDOA.

Note that both E-OTD and U-TDOA struggle in the same 'urban canyons' that GPS struggles with (for the same reasons).

Fingerprinting Systems

For many indoor environments, multipathing is so severe that the multilateration error is too great, or there just isn't hardware with sufficiently accurate timing capabilities available.

In these cases, we may have success turning the location problem into a pattern matching one. The basic idea is to accept that you can't predict what the signal properties will be at a given location based on some signal propagation model, but that some of them will be constant over time (signal strength is a commonly chosen property). So you measure the properties at lots of measured locations in a *survey* and then compute locations by measuring the local signal properties and comparing them to your surveyed 'map' of properties. More formally, there are two phases:

Survey or Offline Phase. You visit each point in a grid covering your tracking area. At each point you measure some location-variant property. When finished, you have a set of points at known locations, each with an associated *vector* of measurements (one for each transmitter).

Tracking or Online Phase. When we want to locate a device, we have it create a vector of measurements in the same way as the survey did. The location task is then to find a grid location with a survey vector that is 'closest' to the measured vector.

0.12.3 Time-Invariance

There is an implicit assumption in all of this that the location-varying property is time-invariant so that a survey performed at 3pm on a Sunday would yield

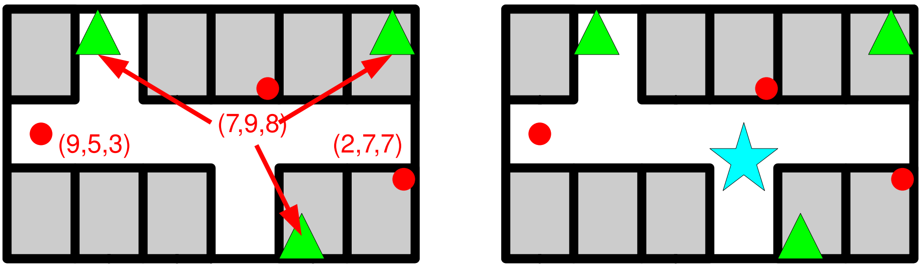


Figure 11: Indoor fingerprinting. Here a basic three point survey is first performed (red dots). Then an incoming measurement (8,8,7) is positioned between the survey points based on some form of interpolation (blue star).

the same results as a survey at 9am on a Friday. This is pretty hard to guarantee (harder as time goes on) and realistically you need to keep updating the survey to account for environmental changes.

Some research systems claim pretty high accuracies, but usually the testing is done in a static environment immediately after the survey. What is needed is a good, long-term study of the accuracy of such systems... Ph.D., anyone?

0.13 Case Studies

0.13.1 The iPhone (Skyhook Wireless)

The iPhone (and the iPod Touch) both use simple WiFi fingerprinting to get coarse location (the iPhone has GPS of course, but that doesn't always work). The phone looks to see which WiFi base stations it can identify in its vicinity and sends this info to SkyHook, who have a database of access points and their locations. They pattern-match across their database and return a position.

How do they have such a database? They employ drivers to drive around countries, with equipment that logs where the car is (GPS) and links it to the results of continual WiFi scans. They rely on most APs being privately owned so that they don't get moved much (and if they do, rarely will they be moved together). The density of APs in a city these days means that it actually works pretty well, as any iPhone user will attest.

In rural areas, of course, there may be few or no WiFi signals, in which case

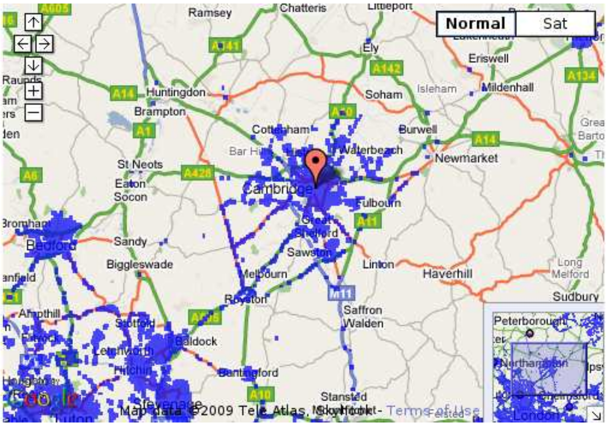


Figure 12: Skyhook's coverage of Cambridge. Each blue dot is a logged WiFi point and position.

you've got no chance (but these areas are, of course, where GPS excels, so...).

0.13.2 Indoor WiFi Systems

Fingerprinting techniques came from researchers who wanted to overlay indoor location tracking on standard infrastructure, particularly WiFi. Initial attempts tried to convert the Received Signal Strength Indicator (RSSI) that WiFi systems report to distance (usually applying a $1/r^2$ law). This sucked because signals get attenuated significantly indoors in wildly unpredictable ways.

The fingerprinting systems were based on surveys where a user moved to a known location and took lots of RSSI readings to as many APs as they could hear. Traditionally, we deploy WiFi APs so that they have some, but not much coverage overlap. Here, we want lots of overlap wherever we are, so we boost the density of APs beyond that required for communication.

A moving WiFi device then takes the resultant WiFi map and locates itself based on what it can observe at any moment, as per Figure 11.

Accuracy

The accuracy is dependent on a great number of factors including stability of the radio environment, time since the last survey, density of APs, etc.

Cutting edge methods apply probabilistic techniques to match a measurement vector to a general co-ordinate position that may lie between survey grid points (basically smart interpolation techniques). These often claim accuracies of sub-metre but this has to be taken with a lorry-load of salt. The statistics are derived from small data sets taken over short timespans in unrealistic environments.

Broadly speaking, positioning accuracies of around 2–5 m seem believable. This is pretty good considering it's being overlaid on an established infrastructure!

Dead Reckoning Systems

0.14 Principles

It's possible to track *relatively* rather than *absolutely*. A relative positioning system gives your location relative to the last location. So, instead of saying 'you are at (x,y,z)' it might say 'you moved 1 m to your left'. Thus we get a stream of incremental position changes that can only be converted to an absolute position if we sequence *all* of them together in order.

A car odometer is a trivial example, assuming the car moves in a straight line. It counts wheel revolutions and adds them together to tell us our (1D) location. Similarly we can integrate the input from accelerometers, use gyroscopes to derive orientation, and countless other techniques.

The big issue with dead reckoning systems is that *errors accrue*. If the tread on your tires drops by 3 mm, the circumference of your wheel is reduced by around 18 mm and after 55 revolutions, the overall distance estimate is out by an entire metre. And if you are double integrating your measurements (as you would when you have accelerations) the positioning error grows with the square of the time!

0.15 Case Studies

0.15.1 XSens IMUs

Recent years have seen an explosion in Inertial Measurement Units (IMUs) based on Micro-Electrical-Mechanical Systems (MEMS). These are mechan-

Figure 13: An XSens MTx IMU

ical systems constructed at the nano-scale and mean we can package up everything we need for an IMU into a small box.

The Dutch company XSens have been quite innovative in this area, and they have a number of matchbox-sized IMUs. We will look at the MTx, $38 \times 5 \times 21$ mm IMU that weighs a mere 30 g (Figure 13). It contains:

- 3D Accelerometers.
- 3D Rate Gyroscopes (to measure angular velocity).
- 3D Magnetometer.

Why all these sensors: can't we just double-integrate the accelerometers? The problem is that the IMU measures the accelerations *in its own frame of reference*. So, yes, we can double-integrate the accelerometers but the distance we get will be meaningless unless you're sure the world axes and the IMU axes didn't change relative to each other. i.e. no rotation of any type occurred. Alternatively, we *measure* the rotation using the gyros and try to compensate. The task becomes:

1. Integrate the angular velocities to get rotations.
2. Resolve the accelerations into the world frame based on the rotations.
3. *Optional.* Use the magnetometer to estimate where magnetic north is to assist the previous step.

4. Subtract gravity from the accelerations.
5. Double-integrate the accelerations to get a position increment.
6. Lather, rinse, repeat.

As you can imagine, the maths isn’t pretty. Worse, we’re integrating all over the place so small errors quickly accrue (they grow cubically in time!!). The biggest problem is usually the gyro integration. Small angular velocity errors quickly build up and the position result is way off.

If you want to make any progress with IMUs, you have to feed in as many constraints as you can. The usual trick to track people is to stick the IMU on the foot. Then when the foot is down, you assert that the velocity of the IMU is zero. This allows you to correct your drift and limits the error growth.

On vehicles, IMUs are often used with GPS systems to ‘fill in’ movements between GPS fixes.

Optical Systems

0.16 Principles

Tracking using optical sensors is no trivial task. Sooner or later we have to be able to map from a pixel to a 3D location so it is constructive to look at how images are formed at the CCD. The basics are illustrated in Figure 14.

Working in 2D for a moment, a point (y,z) in the frame of reference of a cam-

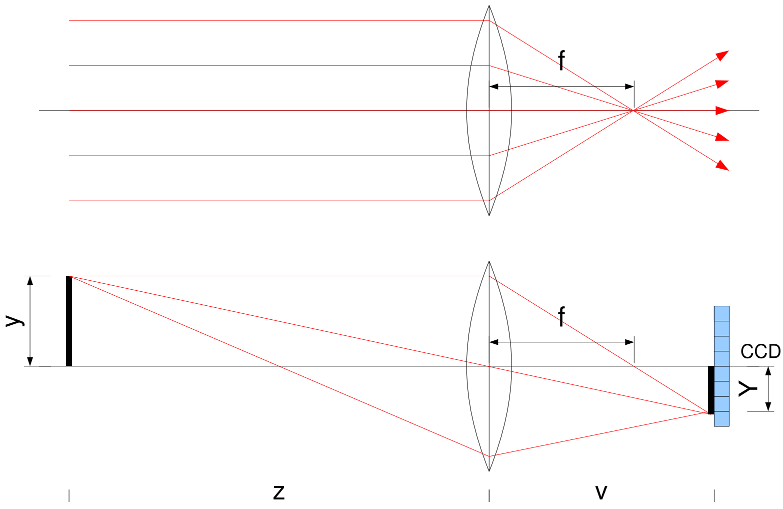


Figure 14: Thin lens optics.

era (z always points out of the camera along the optical axis by convention) will be projected to a point on the CCD that is Y from the centre. From similar triangles we can immediately state:

$$\frac{Y}{v} = \frac{y}{z}, \tag{9}$$

Now, v is not normally known to any accuracy (it's what we vary when we're focusing) but f is an intrinsic property of the lens. We use the lens equation (also trivially derivable from the diagram) to link them:

$$\frac{1}{z} + \frac{1}{v} = \frac{1}{f}. \tag{10}$$

Combining these equations (and adding in the x-direction by symmetry) tells us that any point (x,y,z) in the camera frame will project to (X,Y) in the sensor plane, where:

$$\begin{aligned} X &= \frac{xf}{z-f} \qquad (11)\\ Y &= \frac{yf}{z-f} \qquad (12)\end{aligned}$$

X and Y are distances, and a quantisation turns them into pixel values (effectively multiplying by the pixels per unit distance of the CCD and rounding).

Now, to work this backwards means finding the (x,y,z) point from CCD distances (X,Y). Thing is, we have two inputs (X,Y) and three outputs (x,y,z). Two equations, three unknowns. Not going to work.

What this tells us is that we can't discern point depth from a single camera (you know this just from closing one eye). There are two ways to address this:

Stereoscopic vision. If we have two cameras with known position and pose relative to one another, both of which can see the point source, we can compute (x,y,z) from the measurements set (X_1,Y_1,X_2,Y_2). The maths isn't pretty because we have to work in multiple frames of reference, but the final result is good.

Multiple points with known properties. We can keep a single camera if we have a pair (or more) of points with known properties in the real world. For example, we might know that there are three points in an equilateral triangle of side 10 cm in the real world. If we observe the projections of all three points, we are able to compute the 3D position and pose of the camera relative to it (or vice-versa).

Note that in the latter case, the multiple points define a full frame of reference (unlike a single point) and we must compute the camera position (three variables) *and* the camera pose (the rotational transformations from the camera FoR to the object FoR, another three variables). Thus we find six variables in all, requiring three projected points.

0.17 Performance

Lenses feature varying levels of distortion that must be accounted for based on precise calibration data. This makes everything an order of magnitude harder. However, well configured optical systems are very precise.

In a wide area, it's hard to get complete visual coverage and to calibrate the positions and poses of cameras: small knocks can have big effects on the systems.

0.18 Case Studies

0.19 CCTV

CCTV tracking is an example of unconstrained visual tracking, and is very difficult to accomplish. The biggest difficulty is not usually the conversion of pixels to locations, but rather the identification of the pixels of interest. To track a person is hard because there often isn't a clear notion of a person—just a blob on the screen. But is that blob one person? Or two people? Or one person and a trolley? Or one person and a strong shadow?

Ultimately, unless you can constrain the movements of the people in your scene (turnstiles, lanes, etc.) it's a difficult task to reliable track individuals.

Microsoft Research had a crack at this with their *EasyLiving* project, where they aimed to augment a living room with technologies that could provide ubicomp-like assistance. As part of it, they wished to track the occupants of the space. They used two cameras in a stereoscopic configuration, primarily to make it easier to distinguish the number of people in the room.

Identity was a problem, so they built colour histograms of people and matched incoming screen 'blobs' to them. This made tracking a specific person easier frame-to-frame, but did not of course help if people changed their clothing!

0.19.1 Motion Capture Systems

Motion Capture systems are used extensively in the animation/film industry to provide life-like movements to computer-generated 'actors'. They use an array of cameras carefully calibrated to place them into a single frame of reference. The real actors wear small markers that emit IR (they are either active or reflect IR from lights attached to the cameras). The cameras use IR-filters in front of their lenses to let through the light coming from the markers.

Al the cameras are synchronised to take images at the same instants. For any given round we end up with a set of images with bright white blobs in them. Using stereoscopic vision techniques, the system can match up the blobs and compute an accurate 3D position from them.

These systems can achieve *millimetre* accuracies at high update rates (approaching 1kHz). Unfortunately they are very, very expensive and don't scale well at all. For example, to cover a $2\times2\times2$ m volume would typically require 6–10 cameras all looking in from a few metres away!

0.19.2 Marker-based Tracking

One way you make the optical tracking problem easier is to constrain it to look position specific shapes. In marker-based tracking you use markers or tags (which are just symbols printed on paper—see Figure 15) with known properties to derive enough projected points to figure out the six degrees of freedom (position plus pose).

As an example, Cantag is a framework developed here to do exactly that sort of thing. When it gets an image it:

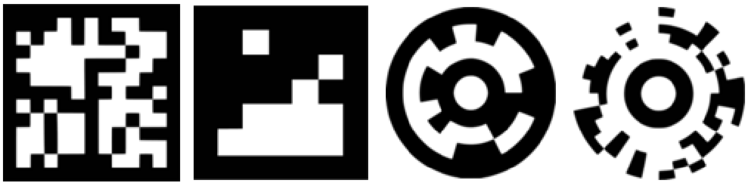


Figure 15: Example tag types from Cantag

- Converts it to monochrome by thresholding;
- Searches for any shapes that could be a projection of what it's looking for (a square tag projects to a quadrilateral so we look for four-sided shapes));
- Uses what it's found to derive 3+ projected points, each of which corresponds to a known feature of the tag (e.g. corners of the square);
- Reverses the projection using information about the tag (e.g. its size) to figure out where the tag is.
- Reads the unique code on the tag (having unprojected it) to figure out which tag it is.

It isn't easy to understand how this might work, so let's look at an example algorithm.

Simple Tag Positioning

We start with two frames of reference. the first is that of our camera (x, y, z); the second is the 'tag' frame (u,v)—there are only two axes because the tags are 2D.

There must be a transformation that gets us from one co-ordinate system to the other, and we can represent it as follows (remember homogeneous matrix transforms from IB CGIP):

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} d_0 & d_1 & d_2 & d_3 \\ d_4 & d_5 & d_6 & d_7 \\ d_8 & d_9 & d_{10} & d_{11} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ 0 \\ 1 \end{pmatrix}, \tag{13}$$

or just

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} c_0 & c_1 & c_2 \\ c_3 & c_4 & c_5 \\ c_6 & c_7 & c_8 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}, \tag{14}$$

where c_n are unknown constants. We measure projected values at the CCD, so we use our projection equations above to say:

$$X = \frac{xf}{z-f} = \frac{f(c_0u + c_1v + c_2)}{c_6u + c_7v + c_8 - f)} \tag{15}$$

$$Y = \frac{yf}{z-f} = \frac{f(c_3u + c_4v + c_5)}{c_6u + c_7v + c_8 - f)} \tag{16}$$

So far, everything we've done is totally correct. But now we're going to try linearising this problem (the equations above are not linear of course). The trick is as follows. Rewrite the above as:

$$(c_8 - f)X = fc_0u + fc_1v + fc_2 - c_6uX - c_7vY \tag{17}$$

$$(c_8 - f)Y = fc_3u + fc_4v + fc_5 - c_6uX - c_7vY \tag{18}$$

and let $a_n = \frac{c_n}{c_8 - f}$ to get:

$$X = fa_0u + fa_1v + fa_2 - a_6uX - a_7vY \tag{19}$$

$$Y = fa_3u + fa_4v + fa_5 - a_6uX - a_7vY \tag{20}$$

Now, Cantag uses squares which means we get four projected points (one per corner). So we actually have four sets of the above two equations, which we can write out as a matrix:

$$\begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = \begin{pmatrix} fu_0 & fv_0 & f & 0 & 0 & 0 & -X_0u_0 & -X_0v_0 \\ fu_1 & fv_1 & f & 0 & 0 & 0 & -X_1u_1 & -X_1v_1 \\ fu_2 & fv_2 & f & 0 & 0 & 0 & -X_2u_2 & -X_2v_2 \\ fu_3 & fv_3 & f & 0 & 0 & 0 & -X_3u_3 & -X_3v_3 \\ 0 & 0 & 0 & fu_0 & fv_0 & f & -Y_0u_0 & -Y_0v_0 \\ 0 & 0 & 0 & fu_1 & fv_1 & f & -Y_1u_1 & -Y_1v_1 \\ 0 & 0 & 0 & fu_2 & fv_2 & f & -Y_2u_2 & -Y_2v_2 \\ 0 & 0 & 0 & fu_3 & fv_3 & f & -Y_3u_3 & -Y_3v_3 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} \tag{21}$$

To solve this we put in the known values of u_n and v_n. We know them because they are just the corners of the square in the tag frame. If we assume the tag length is one unit then the corners in the tag frame are at (0,0), (1,0), (0,1) and (1,1). We use the image to measure (X_0, Y_0), (X_1, Y_1), (X_2, Y_2) and (X_3, Y_3) and then solve the above equation to get the coefficients a_n (this is just matrix inversion and you could use any library you like to do it).

Of course, we wanted c_n and not a_N, and we need to know $(c_8 - f)$ to convert. If you think about it, you'll see that c_8 is the z coordinate of the tag in the camera frame. So we're really being asked to fix the usual distance-projection ambiguity. We assert that the sides of the tag in the camera frame must also be one unit. E.g.

$$\begin{aligned} (u = 0, v = 0) &\quad\Rightarrow\quad x_1 = c_2, y_1 = c_5, z_1 = c_8 \\ (u = 0, v = 1) &\quad\Rightarrow\quad x_2 = c_1 + c_2, y_2 = c_4 + c_5, z_2 = c_6 + c_8 \end{aligned}$$

and we know that $(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2 = 1^2 = 1$ so we get:

$$c_1^2 + c_4^2 + c_7^2 = 1 \Rightarrow a_1^2 + a_4^2 + a_7^2 = \frac{1}{(c_8 - f)^2} \tag{22}$$

Which, given we have estimates for all a_n and we know the focal length of the lens, f, means we can compute c_8 and hence all of the c_n. Which in turn means we have computed the position and pose of the tag relative to the camera.