ES3 Lecture 7

Sensors and location data

Sensors

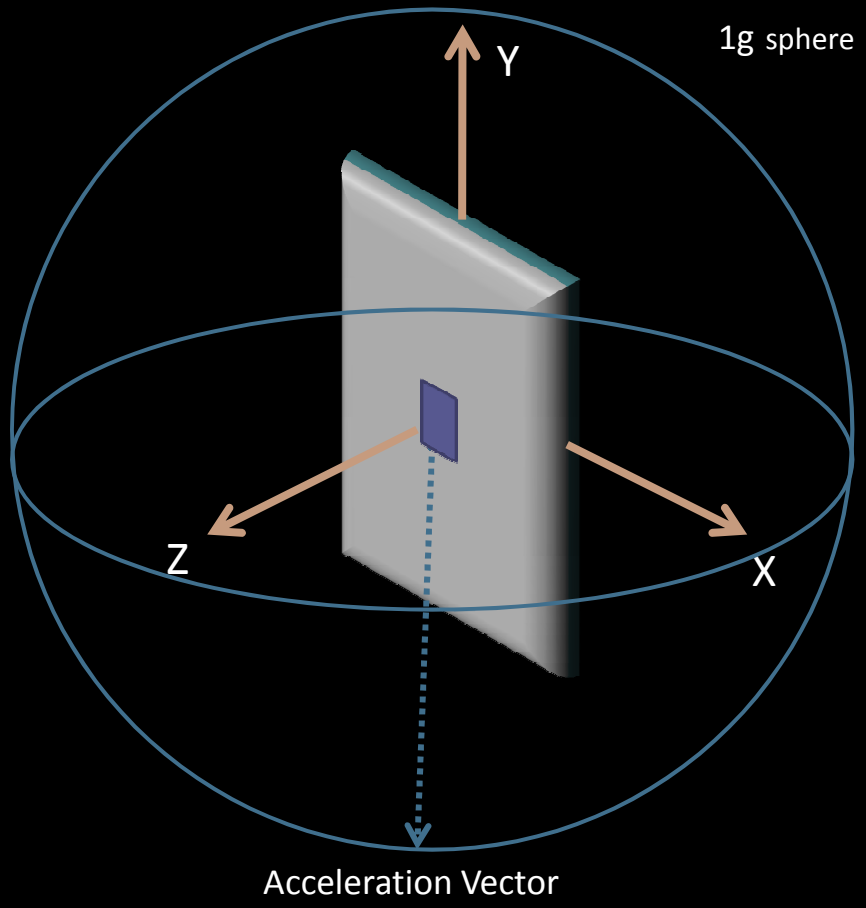
- Mobile devices now often have a wide array of "unusual" sensors
  - Motion sensors (accelerometers, magnetometers, gyroscopes)
  - Proximity sensors (radar, RFID, IR)
  - Cameras, microphones
  - Temperature and environmental sensors
  - GPS and other location sensors (cell towers, wifi triangulation)
  - Pressure sensors and touch sensors
- Lots of exciting interaction possiblities
  - But using these sensors to build interfaces is hard
  - We'll cover some of the low-level techincal issues in sensor-based interaction

Accelerometers

- Accelerometers measure linear acceleration
  - Because of the *principle of equivalence*, accelerometers necessarily measure both acceleration from being moved **and** gravitational fields acting on them
- This means they measure the sum of gravity and movement!
  - They normally measure in units of g
  - 1g = the nominal acceleration induced by the Earth's gravity
- Gravitational acceleration is 1g by definition (varies over the Earth slightly)
  - This is quite large compared to most movements a human makes
  - Human movements have **high instanatenous** acceleration, but **very low average** acceleration
  - Spiky, noisy accelerations

Accelerometers (II)

- Can measure on 1,2, or 3 axes
  - Full 3-axis most useful
- Treat acceleration values as vector
  a =(x,y,z)
- |a| = 1g if at rest (from gravity)
  - |a| = sqrt(x*x+y*y+z*z)
- Angle of orientation:
  - Pitch = atan2(y,z)
  - Roll = atan2(x,z)



Tilt measurement

- Accelerometers can easily be used to measure tilt
  - e.g. the iPhone bubble level
  - innumerable marble-in-a-maze games

- Measure angle relative to gravity
  - Linear accelerations (from moving) have relatively small effect

- Easily make a 2D "mouse-like" control
  - control mapping isn't very good though
    - human hands have limited tilt range and poor accuracy
    - marble games are *hard* in the real world
  - Screen reflection is a massive problem for tilt interfaces

Accelerometer (III)

- Issues:
  - Can't measure rotation about gravity!
  - Sensors not always mounted at centre of gravity
  - Consumes some battery power (continuous processing can be a problem!)
  - Like all sensors, may need calibration
    - y= Ax + B (A matrix of coefficients, B offset vector)
    - Usually just scaling and offseting is required
    - Place device in several static orientations, measure field, compute offset and scale

- Measures a sum of movement and orientation
  - Can detect motion and direction of motion
  - **CANNOT** just integrate up accelerations to get position
  - Variation from slight changes in orientation end up much bigger than any accelerations from linear motion

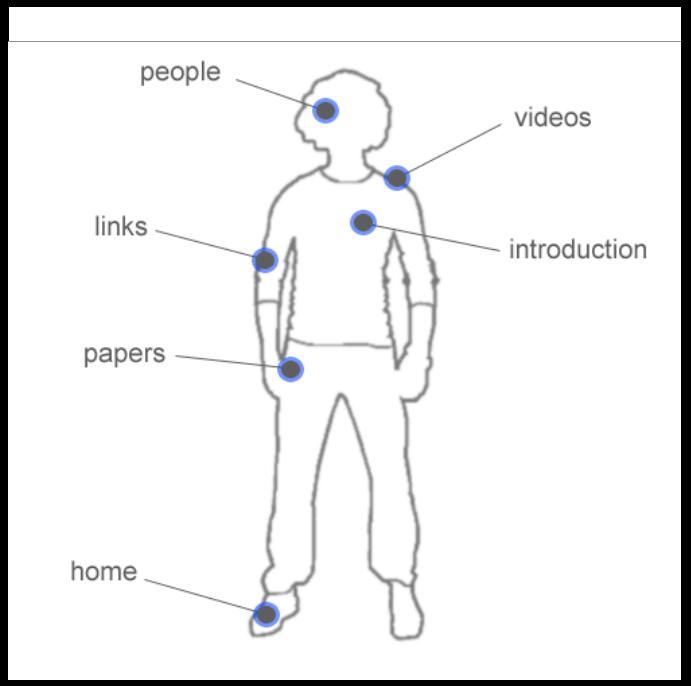
Uses of Accelerometers

- Accelerometer possibilities:
  - Device orientation (landscape/portrait)
  - Shake detection
  - Simple gesture recognition
  - Camera horizon level checking
  - Tilting maze games
  - Tremor detection
- Not possible
  - Position tracking
  - Anything involving measuring rotations about gravity

BodySpace

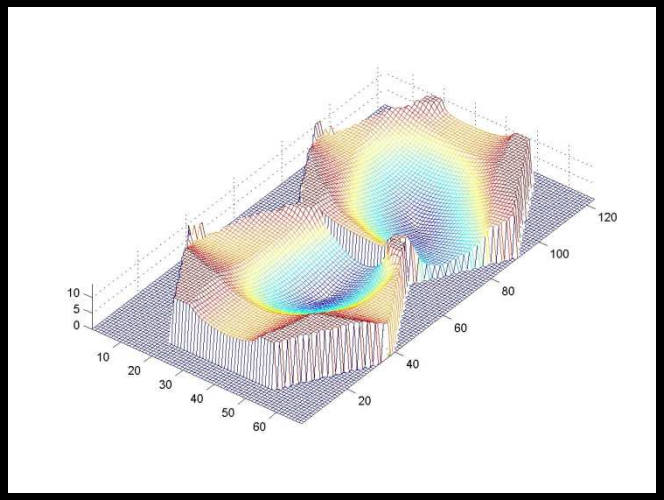
- Track movement to body positions from accelerometer signals alone
  - Look for characteristic movement patterns to that region
  - Note: don't track *position, automatically learn movements!*

- Store data "around the body"
  - a very old mnemonic...



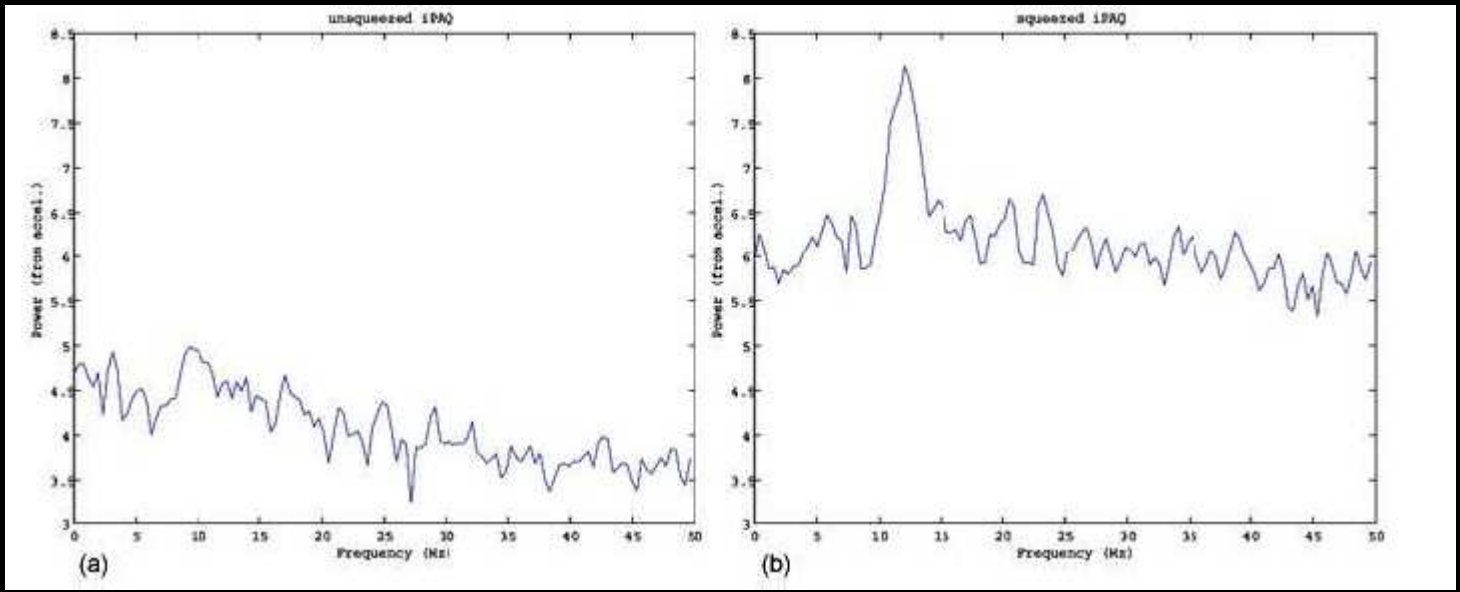
Hex

- Tilt-based text entry
  - Fly through hexagonal tesselation, spelling out words
- Adapts a virtual "landscape" in real time to make it easier to tilt through likely regions



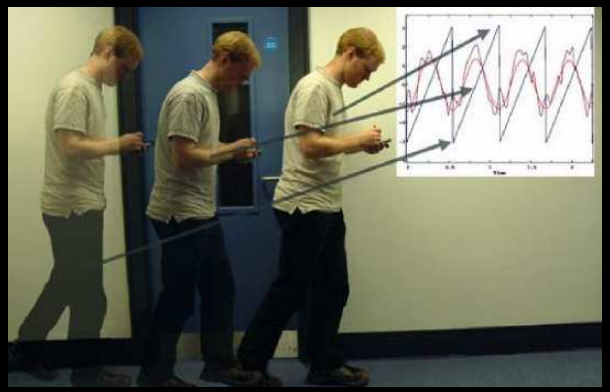
Tremor Detection

- Humans have persistent tremor in the 6Hz--12Hz range
  - Can detect if a human is holding a device by looking for this tremor
- Different poses have different tremor patterns
  - from different muscle groups

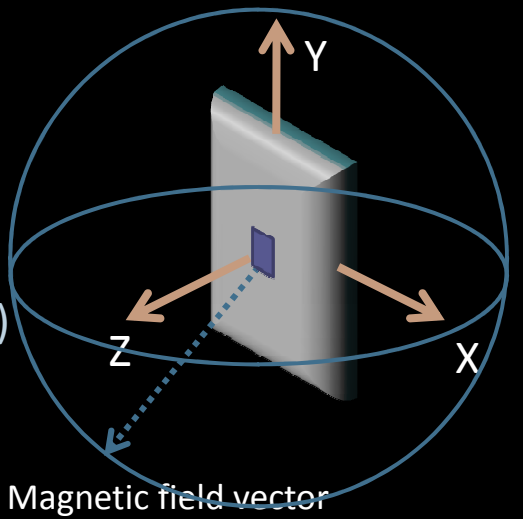


Gait Detection

- Accelerometers can pick up walking motion (for example, if held in hand or in pocket)
  - Track gait patterns
  - Track whether and how fast someone is walking and identify walking style
    - Could be used for limited authentication purposes
- Used in usability analysis to get high-resolution data about disturbances to walking patterns caused by user interfaces
  - Increased cognitive load often leads to breakdown of walking behaviour



Magnetometers



- Magnetometers measure magnetic field strength
  - Measured in Tesla or Gauss (1 microTesla = 10 milliGauss)
  - Primarily the Earth's magnetic field
    - The basis of an electronic compass
- Like accelerometers, usually have 3-axis measurements
  - Note: the magnetic field is a 3D vector -- it doesn't just point North, it points into the ground as well (the **dip angle**)
  - Strength of the field (length of the vector) measures from 300 milliGauss to 600 milliGauss across the Earth
- Strongly affected by local magnetic objects!
  - Do not blindly trust magnetometers

Magnetometers (II)

- Can compute heading
  - Combined with accelerometers, can compute full device orientation
- Must watch out for disturbances!
  - Metal objects, such as building frames, laptops etc. strongly affect the field!
  - Physical range of disturbance roughly proportional to size of object
  - But even tiny objects can have huge deviations at close range
- Check length of the magnetometer vector -- if it's not close to the expected Earth's magnetic field, it's probably inaccurate...
- Regular calibration of the magnetometer readings is essential if location is changing

Edited Operating Instructions

The MK111 Prismatic Compass 1939-1945

Manual supplied with grateful thanks
Dr.A.N. McClean of Bristol

THE WAR OFFICE

26/GS Trg Publications/2121 WO Code No 8868
Manual of Map Reading
Air Photo Reading
And Field Sketching

Section 60 Compass Errors

3 Local magnetic attraction is quite a different thing. It is due to the presence of iron or iron ore nearby. The compass is a sensitive instrument and quite small quantities of iron have a suprisingly large effect on its behaviour. A wristwatch on your wrist when you are using it will throw it out. A steel helmet on your head will cause entirely wrong readings. Steel spectacle frames will affect it. Take the precaution of seeing that anything of the sort is at a safe distance before your start. Small articles will be safe in a trouser pocket, but large articles, such as a rifle, should be two or three yards away. The table below shows the safe distances from various common objects:

Tank	75 yards
Heavy Gun	60 yards
Field Gun	40 yards
Dannaert Fence	10 yards
Steel Helmet	3 yards
Keys/ whistle etc	½yard

4 Iron on the surface can be avoided but there may be iron below ground that cannot be seen. Buried pipelines, shells, mines etc, will all affect the compass if they are close. You may get an obviously incorrect reading, which will warn you that something is wrong but often this error is not big enough to be immediately obvious.

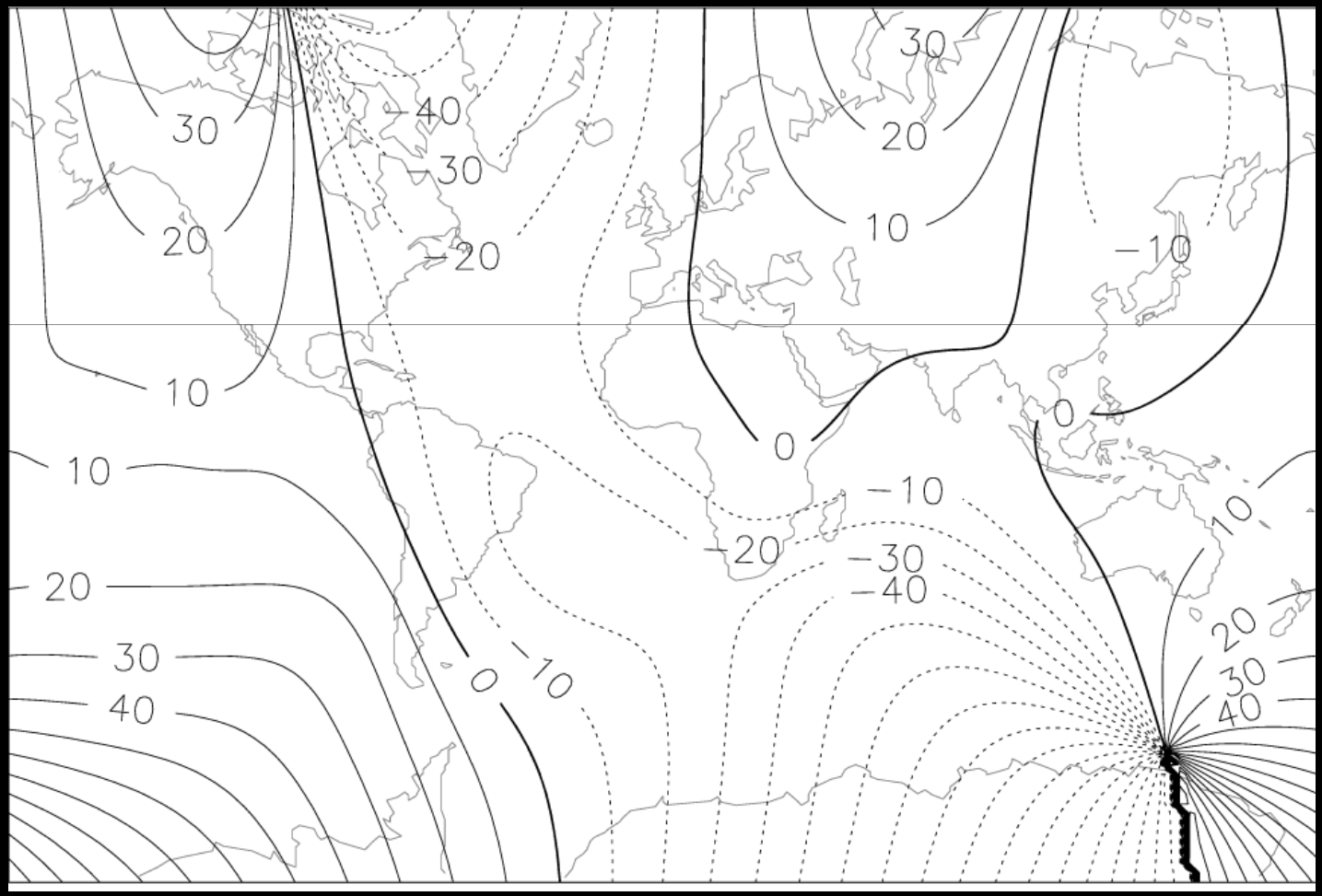
Calibrating a Magnetomter

- Magnetic field should be constant in length at all orientations
  - Can compensate for misscaling of sensor readings by rotating the sensor about and measuring the length of the vector
  - Normalize so the length is == 1 at all angles
- Just get user to wave the device around
  - measure maximum and minimum field values for each axis
- Scale and offset each axis so that the measured values fit in the range -1 ... 1

True North vs North

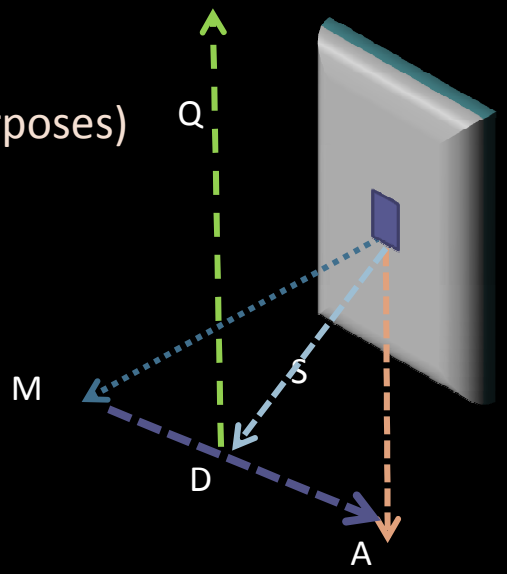
- Magnetic poles are not aligned with Earth's
  - Deviation from true North varies across the Earth
  - At the moment, the poles are at roughly 82'N 118'W and 63'S 137'E
  - Moving all the time
- Need to compensate if getting heading
  - Local field strength and dip angle varies too
- Maps of field variations are available
  - The iPhone, for example, has a field map built in and compenates for magnetic deviations automatically using location awareness

Magnetic field map



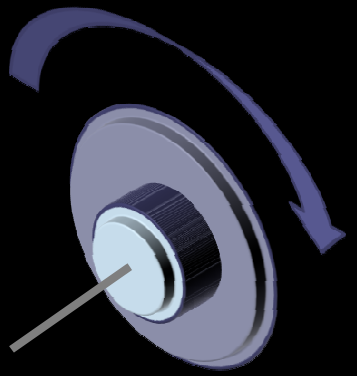
Getting full device orientation

- Between the accelerometer and magnetometer the unambigous orientation of the device can be obtained **(if it is at rest)**
  - Acceleration gives direction of gravity vector (**A**)
  - Magnetometer gives direction of magnetic vector (M)
- These never point in the same direction (for all practical purposes)
- Simple techinque:
  - get average of vectors S=(**A**+**M**)**/2**
  - get difference of vectors D=**A**-**M**
  - get cross product of Q = **S** x **M**



- S, D, Q are three orthogonal vectors
  - can construct a rotation matrix from them just by stacking them together
- Need to align this matrix with real world by calibrating...

Gyroscopes



- Gyroscopes measure *angular velocity*
  - That is, how fast something is rotating around an axis
  - They do not measure orientation!
  - Can be used in combination with acceleration and magnetometers
- MEMS gyroscopes are tiny on-chip sensors
- MEMS gyroscopes often suffer from *drift*
  - Especially from temperature variations
  - These slow variations mean that integrating up angular velocities does not give reliable estimates of orientation
- Fusing together estimates from magnetometers, accelerometers and gyroscopes requires fairly sophisticated algorithms (particle filters, Kalman filters...)

Smoothing and Filtering

- It is usually necessary to filter signals from sensors
  - Sensors often have lots of noise, often concentrated in high frequencies
  - User interfaces using unfiltered signals, for example, will often be very jittery and unusable

- If you want to measure the change in the value of a signal (i.e. its **derivative**), filtering is especially critical
  - The derivative of a signal with high-frequency noise will have **lots** of high-frequency noise
    - To the point that the values are likely to be useless

Filter types

- There are lots of ways of processing signals
  - All reflect **assumptions** about how you believe the underlying value you are trying to measure behaves
  - **They are not magical black boxes**
  - **Bad filtering will make things worse, not better**
    - **THINK FIRST, FILTER LATER!**
  - An accelerometer in the hand moves smoothly because muscles do; so the filtering should reflect that
- Most common filters change the frequency content of a signal
  - e.g. removing high-frequency noise, or low-frequency "drift"
- There are lots of ways of designing and implementing filters
  - We'll cover a very few of the most useful types briefly

Linear filters

- We *always* assume data a series of measurements regularly sampled in time
  - e.g. acceleration at t=0.01, t=0.02, t=0.03 etc.

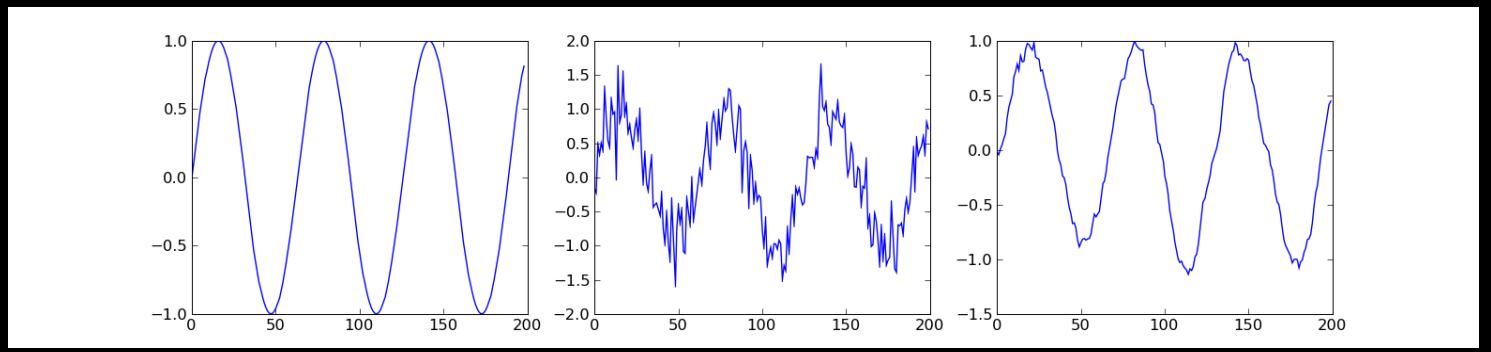
- Most common filters are *linear*
  - They are of the form *y(t)* = a1*x(t-1) + a2*x(t-2) + ... + b1*y(t-1) + b2*y(t-2)
  - i.e. just weighted sums of the previous inputs *and* previous outputs

- The weights a1...an and b1..bn are fixed and define the filter

- Filters are often called FIR if they have only terms involving previous inputs
  - i.e. only ak*x(t-k) terms
- Called IIR if they also have terms involving previous outputs
  - i.e. including bk*y(t-k) terms

Moving Average

- A "moving average" filter just takes the average of a *runninng window* of samples
  - removes high frequency noise
- y(t) = [x(t-n) + x(t-n-1) + ... + x(t)] / n
  - only parameter is **n**
- Simple to implement, but requires storage for previous values
  - This is an FIR filter (no terms involving previous filter outputs)
- Frequency cutoff falls off slowly with increasing **n**



Simple one-pole filter

- A very simple and useful IIR filter is the **one-pole** filter
  - also called an exponential smooth
  - Crude, but often does the job if high-frequency noise needs removed

- Form (x is smoothed value, y is input value)

```
x = alpha * x + (1-alpha) * y
```

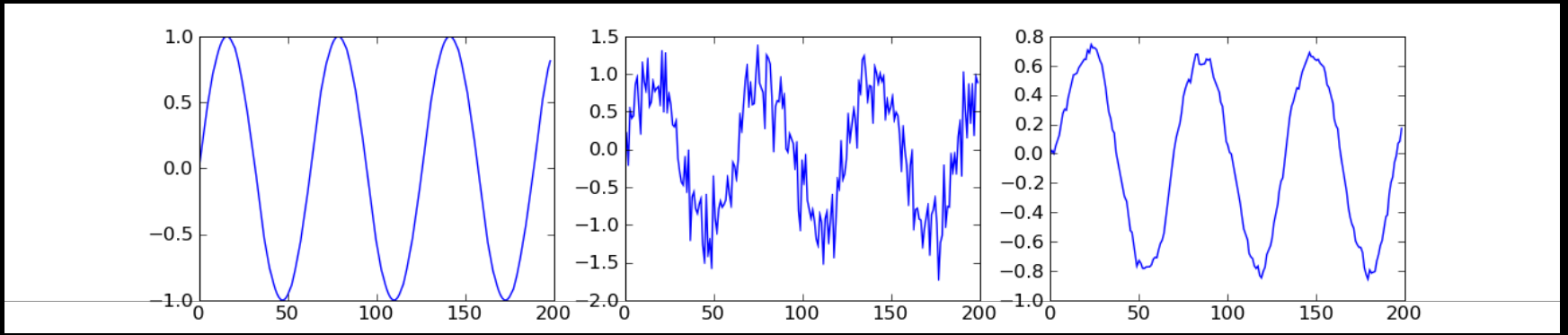
- Only the current value needs to be stored!
- Alpha goes from 0.0 -- 1.0
  - 0.0 is no filtering, 1.0 and **x** never changes
  - Note that **alpha** is very nonlinear 0.7 is much smoother than 0.5 and 0.9 is much much smoother than 0.7

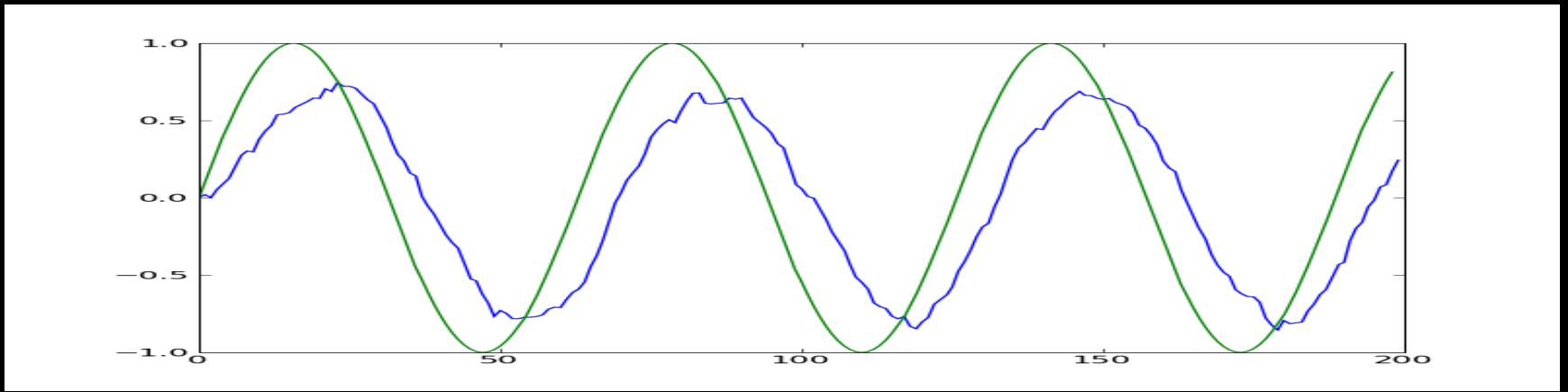
- As a rough guide, the following formula is reasonable way to calculate **alpha**

```
alpha = 1 - (exp(-smoothing)/exp(1))
```

  - **smoothing** goes from 0--infinity

One-pole filter





Designing custom filters

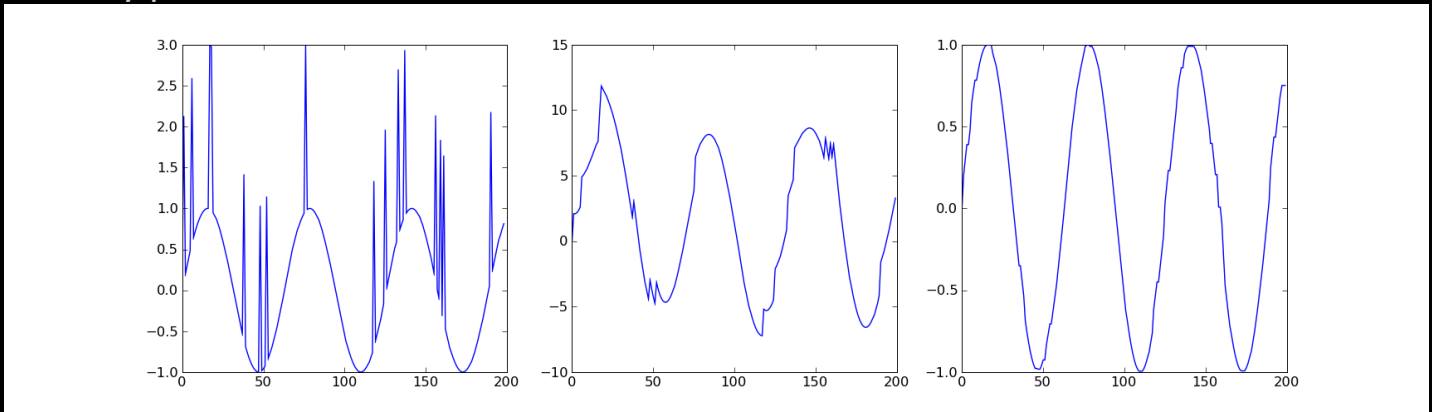
- There are lots of filter design tools
  - normally for designing frequency response of filters
- Usually you can specify the *type*, *frequencies* and *steepness*
  - Types:
    - **lowpass** (remove high-frequencies)
    - **highpass** (remove low-frequencies)
    - **bandpass** (remove everything except for a band of frequencies)
    - **notch** (remove a band of frequencies -- opposite of bandpass*)*

- The frequencies specify when the filter starts cutting out
- The steepness specifies how quickly frequencies are removed
  - Steeper filters remove "more" frequencies but can introduce artifacts

Median Filter

- If the sensor signals are contaminated with *spikes* rather than constant noise linear filters aren't so good
  - They spread out the noise rather than removing it

- Median filters take a *running window* of data and replace the centre value with the median of that window
  - Cleanly removes spikes while preserving other components
- x(t) = median(x(t-n) ... x(t+n))
  - only parameter is **n**



- Other order based filters like the maximum filter, the minimum filter or even a (max-min)/2 filter can be implemented

Physically-modelled interfaces

- One way of designing sensor based interfaces is to *simulate physical properties*
  - Rather than build a complicated recognition system, simulate a mechanical system which can be stimulated
- User learns to *control* the interface based on feedback
  - Feedback is easy to design -- simulate real physical properties
  - Multimodal feedback (audio, vibrotactile, visual...) can be linked to a common *model*
- Information can be encoded in the *model*
  - The user can explore the model by putting energy in

Physically-modelled interfaces

- *Integration* is the key element of physical interfaces
  - Objects in the real world have roughly Newtonian dynamics
  - Force -> acceleration -> velocity -> position
- Interaction is by introducing *forces*
  - Real physical objects can only be affected by forces
  - You can't set an objects velocity!
- Feedback is the result of kinetic energy (i.e. velocity, assuming mass is fixed)
- Simple models, like springs with friction can result in realistic and engaging interactions
  - Look at the "bounce" on iPhone menus, for example

Shoogle

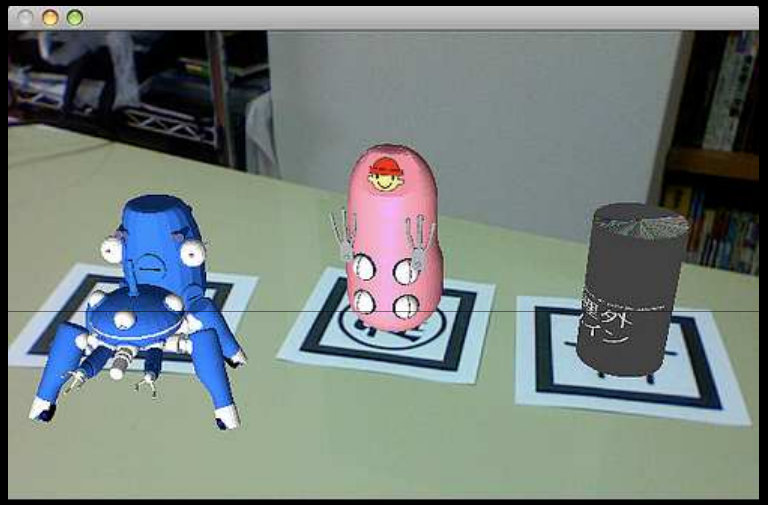
Isomorphism Errors

- Unusual sensing often leads to interaction issues
  - This is a huge research topic -- we can't do it justice here
- One common problem is a mismatch between what a user thinks is being sensed and what actually **is** being sensed
  - People often think motion sensors measure position relative to their local coordinate frame
  - When actually there is often no position sensing, and it is instantaneous motion and orientation which have effects
- Physically-modelled interfaces can mitigate that problem because users focus on *control of the feedback*
  - Not control of the physical world

Camera Input

- Mobile device cameras can be used for input
- Usually either for motion detection or *phicon* tracking
- Motion detection just uses an estimate of the optical flow in the camera frame
  - i.e. how much the whole thing is moving
  - result is a bit like the accelerometer readings
- Phicon tracking uses computer vision to track markers (phicons)
  - These markers are tracked in full 3D position and orientation
  - Can be used for augmented reality purposed by overlaying 3D models on the camera images

Camera Input (II)



Audio Processing: Blowing

- Most mobile devices have microphones
- These can be used for input *beyond* clumsy speech recognition

- Blowing is probably the simplest such use
  - Used on the Nintendo DS for example, and on some iPhone apps

- Very easy to detect
  - Blowing appears as very loud noise
  - Noise is easy to distinguish -- broad frequency spectrum
  - If we have a broad spectrum, total power ~= how hard we are blowing

Audio Processing: Detecting noise

- Simple techinque:
  - use FFT to estimate spectrum
    - can use quite crude spectrum
- Work out standard deviation of frequencies and total power of signal in a running window
  - if standard deviation is high (i.e. broad spectrum) and power is above a threshold, then the power gives the blowing strength

Advanced blowing detection

- Some researchers have extended the techinque using machine learning to classify the noise patterns
  - Can detect *where* around a screen you are aiming your breath with a single microphone!

- A bit crude in terms of resolution, but impressive use of simple sensing hardware
  - BLUI: **low-cost localized blowable user interfaces** Shwetak N. Patel, Gregory D. Abowd, UIST'07