G54MRT Mixed Reality Coursework 2 Introduction

Joe Marshall, Stuart Reeves

The coursework

- Design and prototype a sensor based ubiquitous computing system.
- You have to think of an idea, then:
  - Write a proposal.
  - Send us the proposal to check it and give feedback.
  - Build a working prototype.
  - Test it
  - Write a report and give us the source code and instructions.
- Marks for technical quality, design, testing and report.
  - We **do** run your code
  - Mark scheme in Coursework Spec PDF on moodle.
- Proposal in early -> more time to work on it

How much work is this?

- Half a 20 credit module
- In theory, 72 hours of time on the module is individual work on this coursework.
- There is LOTS of lab time also.
- Several days work programming, plus the lab time, plus time writing the report, reading around the subject etc.
- Enough time to build something meaty
- But don’t go mad
  - Only prototype **part of the system**.
  - Report can discuss how the rest of the system would work (front end, device deployment, form factor etc.)

The Proposal

- Describes the system you want to prototype
- Says what element of the system your prototype will implement (and what technologies, sensors you intend to use).
- A plan for how you will do this in the time available.
  - E.g. say what weeks you plan to do what development in
  - Useful for you to see how you're doing during the project.
- A brief description of how your skills relate to this project
- Template and examples on moodle.
- Deadline: 22nd Feb 15:00
- But get it to us quicker and we'll feedback as quickly as we can, so you can start getting work done.
- **No mark**, but penalty for late / no submission.
- BETTER EARLY THAN PERFECT

Show and Tell

- Half-way **compulsory** feedback session
- You'll get given a 5 minute time-slot in the 29th March lab session
- Some aspect of what you have done should be demo-able, e.g.
  - Show sensor data you've collected
  - Demo an early version of your final system
- We give you feedback
- Be there, ready to demo at your time slot
- If you have to be on main campus directly after, let us know and we'll give you an early time slot.

The Report

- Template document on moodle
  - Summary
  - Background and motivation
  - Related work
  - Design
  - Implementation
  - Testing
- 2000 words = not that many words
  - A report not an essay: factual, concise, detailed.
  - Be visual – use annotated diagrams, graphs, screenshots, photos to make your point.
- Deadline for submission of report and software 13th May 15:00

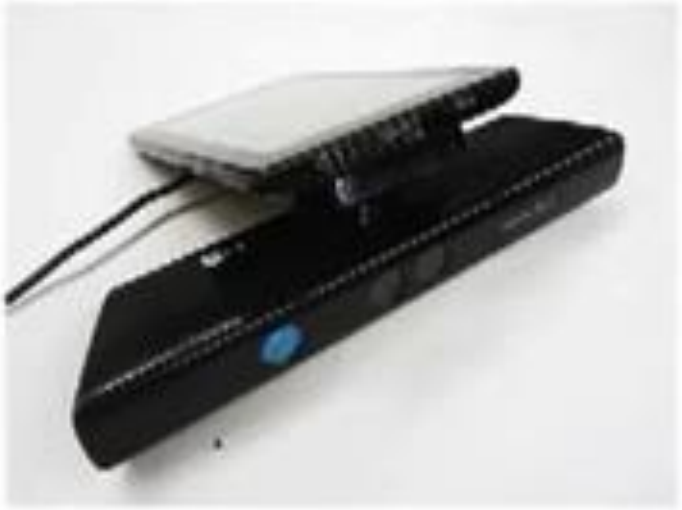
Front End

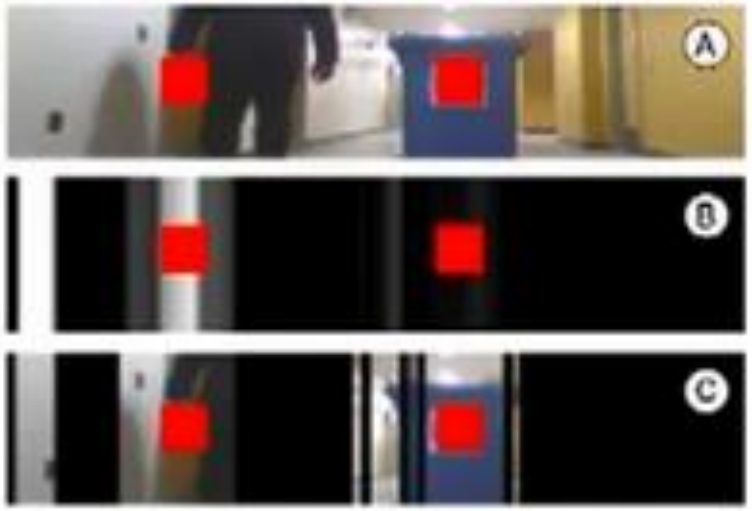
- Fancy front-ends are not the focus of the work on this project.
- The core is the analysis of sensor data.
- You may find a basic front end useful for testing, for getting a handle on what the data looks like, or for generating graphs of data for the report (although you can always bung a text log file into Excel).
- This includes things like sending messages / facebook / tweet / sms whatever.
  - You may not need to actually implement this in your prototype, you can always use something simple like text output to say that at this point it would send an SMS.

We introduce CrashAlert

-
-
-
-

CrashAlert brings peripheral content from the environment into the user's field-of-view, on the mobile device





Where the sensor data comes from

Build a system using the Raspberry PI/GrovePI

- Get the sensor data.
- **Do something clever with it**
- You have to consider: choice of sensors and their characteristics, timing of recording of sensor data etc.

Do something clever with it

- Your project must use at least **two sensors.**
- And do some meaningful processing that analyses or responds to the data from both of them.
- You can't just display sensor data.
- You can (for example):
  - Use sensor data to make inferences about things, e.g.
    - What is the state of the room / environment?
    - What is happening to the system (or an object to which it is attached)?
    - Who is using a system?
  - Use sensor data as an input device, e.g.
    - Where is something being touched?
    - How hard is something hit?
- In the report, you need to demonstrate:
  - What you based your sensor algorithms on (real world data, algorithms from research literature …)
  - What your sensor algorithms do?
  - Do they work? (demonstrate by running tests and reporting numerical results & statistics)

Algorithm Development

- Link it to the real world (during development)
- First record some raw sensor data
- Look at the data (in a text file, in excel, in some other graphing software…)
- Develop an algorithm based on this data (or find an existing one in the literature)
- Show in the report that you have considered the sensor data in your development process.
  - Describe characteristics of the data
  - Put in some graphs showing characteristics of the data

Testing

- The core of this coursework is the development of a sensor algorithm. You need to validate how well the algorithm works by testing.
- Answering the question how well does the algorithm work?
- Numbers are important here:
  - Descriptive statistics – error rates, false positive rates etc.
  - So that it could potentially be compared against other algorithms doing the same analysis.
- You may find that your algorithm doesn't work very well.
  - If you a)do the tests, and b)characterise the accuracy of your algorithm well in the report, a poor algorithm may do better in this coursework than a good algorithm which is not tested.
  - So please test, and test lots.

FAQs from previous years

- Can we work together
  - Marked separately
  - Must both build different systems
  - Your systems may communicate
  - Both systems must use sensor data as per the spec
- Can we write a phone app / web site / java app etc.
  - No
  - Whatever you build **must** use sensor data from the raspberry pi sensors, by having a PI involved
  - If you really want, you can run a webserver on the PI or whatever.
  - But it is a pain, I wouldn't recommend it, and you don't get more marks for a fancy front end.
- Do I have to program
  - **YES!**
  - If you hate programming, you may want to emphasise data capture, analysis and testing in your project, and use a relatively simple algorithm.
  - But we will expect to see your code, and will run your code in marking.

Two examples

- These are two ideas from previous years.
- First, one which is sensing **what is being done to** the sensing device.
- Second, one which is sensing **what happens in the environment of a fixed sensor**.

Sensing Actions: TennisSense

- Detect strokes being played in a game of tennis by attaching sensors to a racket.

TennisSense: Proposal

- A full one is on moodle

Summary

I will build a system to detect racket strokes and ball hits in a game of tennis. This could be used for example in training, to look at how many strokes you'd taken, whether you miss more strokes backhand or forehand.

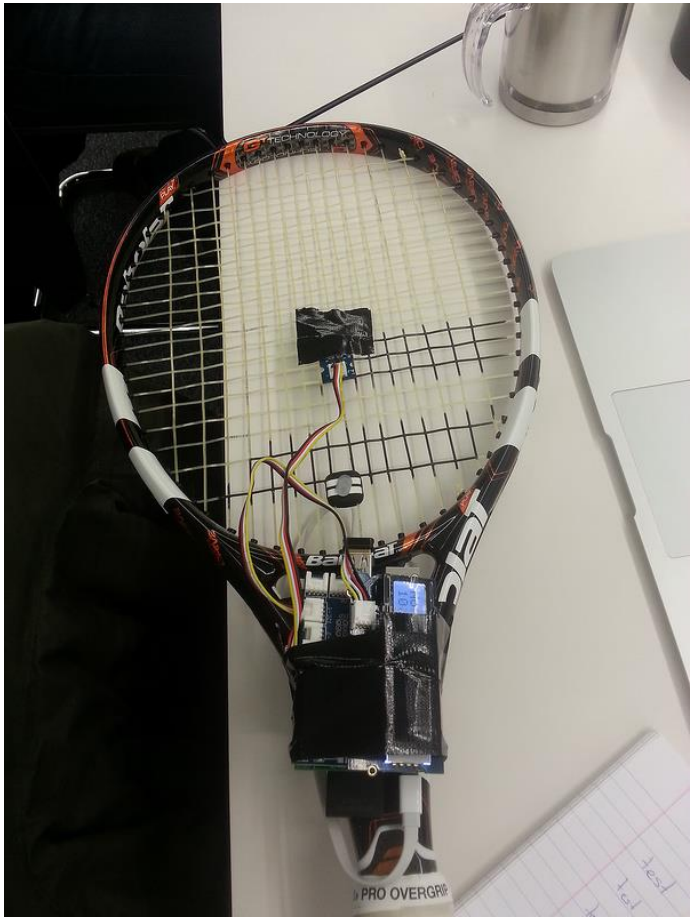
Technologies and sensor data

I will use accelerometer to detect swinging of the racket, and a sound sensor to detect the ball hitting the strings.

…

TennisSense : prototype

- A load of sensors strapped onto a tennis racket.
- Accelerometer to detect swing (in the racket)
- Sound sensor on the strings (placement?)
- Everything strapped onto a tennis racket
- Note: you wouldn't play a real game of tennis with this prototype, but it is enough to demonstrate the sensing (which is what we care about)



TennisSense

- During development, you are likely to have to questions as to how to design your algorithm, e.g.:
  - How can I filter the accelerometer data to correctly detect when someone swings and ignore other events?
  - How can I tell the difference between forehand and backhand swings?
- Ways to answer these questions:
  - Record sensor data from a set of swings and hits and continually tweak the algorithm.
  - Test the algorithm with a new set of swings / hits and see how well it works.

TennisSense: Test 1

- Test the hit sensing separately.
- Do a series of ball hits, count what percentage of hits are correctly registered, also what number of false ball hits there are.
- Plot true positives, false positives and false negatives.
- Look at raw sensor data to see why the errors are occurring.
- You could also individually test the swing sensing.

TennisSense: Test 2

- 'Integration testing' – testing full system
- Do a series of swings of the racket, some accompanied by ball hits, while recording system output.
- Look at whether a)swings and hits were detected at all, b)the correct type of swing was detected.
- Look at raw sensor data to see why errors are occurring.
- Consider whether the sensing of one thing affects sensing of the other…

Fixed Sensors: Seismo-sensor

- A seismic sensor (earthquake detector) for use in areas where radio signal wouldn't work
  - High EMI areas
  - Underwater
- Uses accelerometer sensor to detect seismic waves (vibrations)
- Needs to synchronise multiple sensor instances to detect position. Timing of waves must be exact.
  - One way to do this is via line of sight laser transmission.
  - Uses a light sensor to detect 'laser pulses'
- Requires testing to demonstrate it works.
  - Generate 'fake earthquakes'
  - Demonstrate time synchronisation.

Seismo-sensor: Proposal

Summary

A set of sensors for detecting seismic vibrations (ground movements due to earthquakes etc.) in areas where radio signal is not practical such as in areas with high electro-magnetic interference or underwater. They will record seismic vibrations, and store this. The sensors need to be time synchronised in order to allow triangulation of the source of the vibration. This will be done by putting the sensors within line of sight of each other and using visible light pulses (lasers in the real design) to synchronise clocks.

Technologies and sensor data

I will use the accelerometer to detect the vibrations. I will also use the light sensor in order to detect light pulses in order to synchronise the clocks. I will use GrovePI boards to build this.

Seismo-sensor: Prototype

- Uses light sensor to detect a ‘laser pulse’
  - In prototype, can be a manually activated torch or an LED connected to a PI or something, we’re not letting you loose with real lasers.
- Synchronises clocks on detected light pulse, and records accelerometer data time-stamped relative to light pulse time.
- Processes sensor data to detect vibration waves characteristic with seismic events.
- Technically difficult – challenges of timing calculation & synchronisation, how the accelerometer data is processed into events, getting a reliable light pulse detector.
  - Sensing and testing of it can be the whole of the prototype, with output written to a text file or similar.
  - Wider system design could be described in the report
    - How the data would be brought together and displayed to users
    - How the sensors would be deployed

Seismo-sensor: Testing 1

- Test 1: Does the sensor detect vibrations
  1. Put a single sensor node on table
  2. Wobble the table several times, noting when you wobbled.
  3. Look at the recorded data, did the program detect vibrations?
  4. For the report, graph the raw accelerometer data and label detected vs non detected vibrations, record what percentage of vibrations were correctly detected, how many false positive vibration detections occurred.

Seismo-sensor: Testing 2

- Test 2: Does the sensor time synchronise correctly
  1. Put two sensor nodes on a table right next to each other.
  2. Put the two light sensors next to each other
  3. Synchronise the timing by shining a light on the two sensors
  4. Wobble the table, pause for a few minutes, then wobble again.
  5. Repeat a few times, with and without synchronisation pulses.
  6. With the recorded data: Did both nodes detect the same vibrations? If not, what percentage were recorded differently? When both nodes detected a vibration, did they detect it at the same time, or how close to the same time were they?
  7. For the report, graph the acceleration data as recorded by each node and mark the detected / non detected vibration times. Present statistics on the mean and standard deviation of the timing error, try and calculate how much timing drift there is over time, e.g. You could say: *Directly after light sensor synchronisation, mean timing error is 0.001 seconds (s.d. 0.002s), this error increases by 0.1 seconds per 10 minutes (at 10 minutes: mean 0.011, s.d. 0.05s).*

What kind of thing might you do?

- Two general areas you might want to work on
- Not worked through ideas
- Lots of different things could be done in these areas.
- Project ideas document on moodle has this same list with a lot more detail

Fixed sensing (i.e., stationary sensing device(s))

- Primarily about interpreting data that is being collected from a particular place or set of places in which sensors are installed. A simple real world example is the Nest thermostat which—based on several basic sensors: temperature, motion and humidity—decides the best temperature for your home based on presence, weather data etc.

Fixed sensing ideas

- Augmenting a fixed object, e.g. coffee machine:
  - Recognising who is using the machine and dispensing their favourite coffee automatically
  - Keeping count of how many cups a person has each day and when they have them
  - Emailing the servicing contract holder when the machine requires maintenance
  - Recording who does maintenance for the machine (e.g., user-led cleaning) and who ignores these tasks
- Augmenting an activity / space:
  - Augmenting an activity that happens in a particular place.
  - Example in the document of augmenting a bathroom to encourage good hygiene practices (washing hands)
- Door Counter
  - How many people are going through a door
  - This is surprisingly hard (you won't do this perfectly)
  - Thinking about constraints on the door position, location etc can make this easier.
- Table Piano / table drums
  - Use sensors to detect touches on a table to trigger drum sounds – turn any surface into a piano / drum machine.
  - What sensors? Someone did this last year with sound sensors to detect hits plus ultrasonic rangers to detect hand position.
  - Don't worry about actually outputting audio, we don't care.

Mobile Sensing (sensing device(s) can move)

- Build a device that prototypes a sensor platform that is mobile in some way. At its simplest level this might involve using the Raspberry Pi as a way to add more advanced sensing to some existing mobile device (e.g., a phone or tablet). Alternatively you can use the Raspberry Pi as a way to prototype some other device which involves mobility.

Mobile sensing examples

- Advanced sensing for mobile phones
  - What could you do if a mobile device had extra sensors?
  - E.g. ultrasound, temperature…
- Sports and fitness devices
  - Using sensors in sports timing applications (e.g. finish line timing, lap counting, ‘how extreme is my downhill mountain bike ride’)
  - Using sensors to detect quality of sports actions (e.g.golf swing quality)
- Augmenting a ‘thing’ / non-fixed object
  - prototype a device that would be attached to an object that moves around space in some way.
  - Parcel tracking with information about how it is handled.
  - Augmented bike frame, performs a set of useful tasks for the cyclist e.g., behaving as an alarm as well as a trip tracker.
  - …

Summary

- Think up an idea
- Write a proposal
- Proposal deadline is 22nd Feb

Dates and lots of time for questions

- Proposal deadline: 22th Feb 15:00
- Show and Tell feedback session:
  29th March (in lab session)
- Final report and software deadline:
  13th May 15 :00 PM
- Any questions, ask now, or come up and ask