DEPARTMENT OF COMPUTER SCIENCE

Automatically Correcting for a Time Drift in Accelerometers using RGBD C

Connor Williams

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Bachelor of Science in the Faculty of Engineering.

Wednesday 20th April, 2016

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of BSc in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Connor Williams, Wednesday 20th April, 2016

Contents

Abstract

Obesity, depression, stroke, falls, cardiovascular and musculoskeletal disease are some of the biggest health issues and fastest-rising categories of healthcare costs in the UK. The associated expenditure is widely regarded as unsustainable and the impact on quality of life is felt by millions of people in the UK each day. With a rapidly ageing population - could technology be the answer to some of these problems?

SPHERE (a Sensor Platform for HEalthcare in a Residential Environment) is developing a number of different sensors that will combine to build a picture of how we live in our homes. This information can then be used to spot issues that might indicate a medical or well-being problem.

The technology could help by:

- Predicting falls and detecting strokes so that help may be summoned.
- Analysing eating behaviour - including whether people are taking prescribed medication.
- Detecting periods of depression or anxiety.

[1]

Within the SPHERE research group, one of the sensors being developed is a wearable accelerometer. This accelerometer induces a drift between the true time and the local time on the device large enough that the wearable needs to be re-synchronised for every use.

This document looks to consider video data from RGBD cameras around the house, which provides data on where the person is, together with the acceleration data from the wearable sensor. With this data it should be possible to design an algorithm which automatically corrects the time drift so that the accelerometer and the video remain synchronised.

This project has been broken down in the following steps:

1. Consider different types of temporal distortions which could take place on a device such as the one in question.
2. Implement a program which generates synthetic data and can apply any combination of the different types of temporal distortion.
3. Devise a method to compensate for the different temporal distortions.
4. Given a synchronised set of data from the accelerometer and the cameras, introduce artificial drifts between data sets and assess how well the method work.
5. Apply this method to real drifted data which has already been collected in the SPHERE house in three locations; hallway, living room and kitchen.

Implicit to all of these steps was a large amount of data formatting and visualisation.

Supporting Technologies

- Python for implementation.
- SciPy: Numpy, Matplotlib.... for help.

Notation and Acronyms

SPHERE	:	a Sensor Platform for HEalth in a Residential Environment
RGB-D Camera	:	ASUS Xtion PRO cameras which can track colour as well as depth.
Accelerometer	:	A three axis accelerometer on the dominant wrist, attached using a strap
Temporal Distortion	:	A discrepancy in the timestamp of the data
No Distortion	:	No discrepancy
Constant Distortion	:	Clock constantly offset from the ground truth clock
Linear Distortion	:	Clock slower or faster than the ground truth clock
Periodic Distortion	:	Clock sometimes is faster, sometimes is slower than the ground truth clock according to a s
Triangular Distortion	:	Clock sometimes is faster, sometimes is slower than the ground truth clock according to a t

Acknowledgements

Chapter 1

Contextual Background

Obesity, depression, stroke, falls, cardiovascular and musculoskeletal disease are some of the biggest health issues and fastest-rising categories of healthcare costs in the UK. The associated expenditure is widely regarded as unsustainable and the impact on quality of life is felt by millions of people in the UK each day. With a rapidly ageing population - could technology be the answer to some of these problems?

SPHERE (a Sensor Platform for HEalthcare in a Residential Environment), a partnership between University of Bristol, University of Reading and University of Southampton, is developing a number of different sensors that will combine to build a picture of how we live in our homes. This information can then be used to spot issues that might indicate a medical or well-being problem.

The technology could help by:

- Predicting falls and detecting strokes so that help may be summoned.
- Analysing eating behaviour - including whether people are taking prescribed medication.
- Detecting periods of depression or anxiety.

[1]

SPHERE will work with clinicians, engineers, designers and social care professionals as well as members of the public to develop these sensor technologies, making sure that the technology is acceptable in people's homes and solves real healthcare problems in a cost effective way. The SPHERE project also aims to generate knowledge that will change clinical practice, achieved by focusing on real-world technologies that can be shown working in a large number of local homes.

Within the SPHERE research group, one of the sensors being developed is a wearable accelerometer. It was decided that an accelerometer would be useful for monitoring health because... Currently, this accelerometer induces a drift between the true time and the local time on the device which is large enough that the wearable needs to be re-synchronised after every 20 minute use which is clearly not ideal. This document looks to consider video data from the RGB-D cameras around the house, which provide data on where the person is, together with the acceleration data from the wearable accelerometer. With this data it should be possible to design an algorithm which automatically corrects for the time drift so that the accelerometer and the video data remain synchronised.

1.0.1 Sensors and the Smart Home

Currently all sensors in the home are synchronised with NTP (Network Time Protocol) however the procedure that is currently in place is infeasible for real deployments because...

Accelerometers

Participants wear a sensor equipped with a tri-axial accelerometer on the dominant wrist, attached using a strap. The sensor wirelessly transmits data using the BLE (Bluetooth Low Energy) standard to several receivers positioned within the house. The outputs of these sensors are a continuous numerical stream

of the accelerometer readings in units of g. Accompanying the accelerometer readings are the RSSI (Received Signal Strength Indications) that were recorded by each access point. The accelerometers record data at 20Hz, and the accelerometer readings range is 8g. RSSI values are also recorded at 20 Hz, and values are no lower than -110 dBm.

Due to the nature of the sensing platform, there may be missing packets from the data.

RGB-D Cameras

Video recordings are taken using ASUS Xtion PRO RGB-D cameras. Automatic detection of humans is performed using the OpenNI library, and false positive detections were manually removed by the organizers by visual inspection. In order to preserve the anonymity of the participants the raw video data are not shared. Instead, the coordinates of the 2D bounding box, 2D centre of mass, 3D bounding box and 3D centre of mass are provided. The units of 2D coordinates are in pixels (i.e. number of pixels down and right from the upper left hand corner) from an image of size 640480 pixels. The coordinate system of the 3D data is axis aligned with the 2D bounding box, with a supplementary dimension that projects from the central position of the video frames. The first two dimensions specify the vertical and horizontal displacement of a point from the central vector (in millimetres), and the final dimension specifies the projection of the object along the central vector (again, in millimetres).

RGB-D cameras are located in the living room, hallway, and the kitchen. No cameras are located elsewhere in the residence.

The current solution for avoiding this time drift affect the SPHERE group is to synchronize the sensor every X minutes?

This algorithm could more generally be used to synchronize two 1-dimensional signals such as voice recordings.

If the algorithm as a result of this project is able to correct for the drift of the SPHERE data, it will allow for advancements in the SPHERE project such as:

1. Detecting strokes????

Challenges involved in this project are:

1. NTP is infeasible because...
2. Dont know the current time drifts.

The high level objective of this project is to create an algorithm which can synchronize a 1-dimensional signal A with a ground truth signal B given that signal A has been affected by one of the following temporal distortions:

- No Distortion
- Constant offset
- Linear Distortion
- Periodic Distortion
- Triangular Distortion

This project could be broken down in the following steps:

1. Consider different types of temporal distortions which could take place on a device such as this one.
2. Implement a program which generates synthetic data and can apply any combination of the different types of temporal distortion.
3. Devise a method to compensate for the different temporal distortions.

4. Given a synchronised set of data from the accelerometer and the cameras, introduce artificial drifts between data sets and assess how well the method works. This may reveal new information about the SPHERE data which could influence the future of the project.

Implicit to all of these steps is a fair amount of data formatting and visualisation. I used Python for the implementation with SciPy which is an open source library of scientific tools for Python.

Chapter 2

Technical Background

2.1 Synchronising two device clocks

One way to interpret the problem at hand would be to treat it as an issue of finding a ground truth time when two clocks are running out of synchronisation from each other.

In a system where all of the sensors were connected to a central device the solution is trivial; the centralized device will dictate the system time. Cristian's algorithm and the Berkeley Algorithm, described below, are some solutions to the clock synchronization problem in a centralized server environment.

In a distributed system, such as the one we have, the problem takes on more complexity because a global time is not easily known. The most used clock synchronization solution on the Internet is the Network Time Protocol (NTP) which is a layered client-server architecture based on UDP message passing.

2.1.1 Cristian's algorithm

Cristian's algorithm relies on the existence of a time server. The time server maintains its clock by using a radio clock or other accurate time source, then all other computers in the system stay synchronized with it. A time client will maintain its clock by making a procedure call to the time server. Variations of this algorithm make more precise time calculations by factoring in network radio propagation time.

2.1.2 Berkeley algorithm

The Berkeley algorithm is suitable for systems where a radio clock is not present, this system has no way of making sure of the actual time other than by maintaining a global average time as the global time. A time server will periodically fetch the time from all the time clients, average the results, and then report back to the clients the adjustment that needs be made to their local clocks to achieve the average. This algorithm highlights the fact that internal clocks may vary not only in the time they contain but also in the clock rate.

Often, any client whose clock differs by a value outside of a given tolerance is disregarded when averaging the results. This prevents the overall system time from being drastically skewed due to one erroneous clock.

2.1.3 Network Time Protocol

Network Time Protocol a class of mutual network synchronization protocol that allows for use-selectable policy control in the design of the time synchronization and evidence model. NTP supports single inline and meshed operating models in which a clearly defined master source of time is used ones in which no penultimate master or reference clocks are needed.

In NTP service topologies based on peering, all clocks equally participate in the synchronization of the network by exchanging their timestamps using regular beacon packets. In addition NTP supports a unicast type time transfer which provides a higher level of security. NTP performance is tunable

based on its application and environmental loading as well. NTP combines a number of algorithms to robustly select and compare clocks, together with a combination of linear and decision-based control loop feedback models that allows multiple time synchronization probes to be combined over long time periods to produce high quality timing and clock drift estimates. Because NTP allows arbitrary synchronization mesh topologies, and can withstand (up to a point) both the loss of connectivity to other nodes, and "falsetickers"that do not give consistent time, it is also robust against failure and misconfiguration of other nodes in the synchronization mesh.

NTP is highly robust, widely deployed throughout the Internet, and well tested over the years, and is generally regarded as the state of the art in distributed time synchronization protocols for unreliable networks. It can reduce synchronization offsets to times of the order of a few milliseconds over the public Internet, and to sub-millisecond levels over local area networks.

A simplified version of the NTP protocol, SNTP, can also be used as a pure single-shot stateless master-slave synchronization protocol, but lacks the sophisticated features of NTP, and thus has much lower performance and reliability levels. [**?**]

There are also other clock synchronisation methods over connected devices such as Clock Sampling Mutual Network Synchronization, Precision Time Protocol, Synchronous Ethernet, Reference Broadcast Synchronization and even Global Positioning System.

2.2 Synchronise two 1-dimensional signals along the time axis

The way which I have found a solution to this problem is by synchronising the data after it has been collected.

- Can derive acceleration from position.
- Show derived acceleration from video data is highly correlated with accelerometer data when time is not drifted. This justifies this way is the right way.
- Example with simple data.
- How does one track a drift?
- What does 'drift' even mean?
- Dynamic Time Warping - why won't it work?
- Cross correlation - why won't it work?
- Sliding window cross sorrelation - how might it work? pseudo code?
- Periodic data - how do we avoid cyclic maximums of cross correlation?

Chapter 3

Project Execution

3.1 Data Generator

This section is about my data generator:

- Why is it necessary?
- What features does it have?
- Why does it have these features? (Linked to different temporal distortions)

3.2 Methods

This section will be about the methods I have found which solve the different types of temporal distortion.

- What are the methods?
- How well do the methods work?
- Do other temporal distortions affect the performance?
- What combination of distortions are amendanble?

The project began with me being supplied with a set of synchronized data which had been collected from the SPHERE house. My initial task was to figure out what the data contained. At this point I created a repository on GitHub and a CS Blog which Peter and Niall could subscribe to. The data contained three different files which could be of interest to me:

- Video data: Frame index and timestamp.
- Skeleton data: Frame index and user's 3D joint positions.
- Accelerometer data: Timestamp and X,Y,Z acceleration.

I first decided to visualise the data, looking for properties which could be useful for synchronisation, for example periodic sections of data or extreme movements. I was also supplied with a program which had previously been used to visualise skeleton data from a Kinect camera. I have edited this program in order for me to be able to visualise the skeleton data I have.

The next step included some data formatting and pre processing of the data:

- Removed accelerometer data which is collected before a skeleton is detected.
- Extracted data segments between two time points.
- f(x) = position, f'(x) = velocity, f"(x) = acceleration

- Differentiate video data twice to obtain velocity and acceleration.
- Integrate accelerometer data twice to obtain velocity and position.
- Plotted sub-sequences of data in different ways to confirm correspondence of the data. Different joints, components and differentials of the data returned much different visualisations which made it difficult to decide which would be best to use in general.

At this point I felt that I understood the problem enough to begin to look at similar problems and papers which could help me move towards a solution. It initially seemed like a trivial problem or at least one which must have come up before. Surely device clocks always run out of sync? I also naively assumed a time drift to simply mean one clock is running faster than the other. I came across loads of papers which solved the problem of device clocks running at different rates but all used networks or data from other hardware components like gyroscopes, which would not work for the devices and data we have.

It seemed that a few papers mentioned cross-correlation for synchronising signals and tracking shifted data. For this reason I researched what it was and how (if at all) it could be used to track the drift. It could be used 'out of the box' to detect a shift, but not a drift due to the fact that a drift is a shift over time. I proceeded to plot the cross-correlation between the accelerometer and skeleton data to see if I could observe anything useful, however at this stage I came across a few difficulties:

1. The video data is extremely noisy and differentiating it results in amplified noise. Can we use smoothing or do we need to tackle the problem without differentiation?
2. The video data has a huge feature set after the data formatting and pre-processing where I calculated velocity and acceleration. There are 10 joints which may all be correlated in some way to the acceleration of the dominant wrist, and each joint has 3 dimensions. There is a decision to be made to choose which one(s) to use?
3. Literature on similar problems claim that PCA is required to reduce dimensionality and is useful when compensating for the accelerometer rotating around an axis.

Because of these complications, I decided to try and ignore the complexity of the data for the time being by creating my own data and starting from a simple data set. I generated a sine wave which represents one feature from the video data e.g. the X position of the right wrist. I have calculated the second derivative of this sine wave which in this example represents be the X acceleration of the right wrist. I have then added a time drift to this acceleration data.

At this point I started to notice that there are many variables which could affect a device clock and time drifts could actually be a lot more complicated than I first thought. Things which affect clock speed are:

From here, I plotted the cross correlation of this data which, as mentioned before, could not be used on it's own to track the drift. Because a drift is a shift over time, the cross correlation of a small window of data could be used to correlate that window. Building on this idea, I plotted a sliding window cross-correlation which can detect the lag at every time window. If the data is periodic, this window size must be smaller than the time period.

The plot of this data was a lot more interesting and I was visually able to track the drift. There was however an issue that because the data is periodic, the cross correlation of a window could be maximal at different lags. I needed to figure out how to solve this problem. One simple but not ideal way to do it was to say if the time drift suddenly jumps from a high number to zero, this is the case where we are getting a cyclic maximum x corr.

Chapter 4

Critical Evaluation

Exactly what can my algorithm do? Show it working with different distortions.
Can it do more than described in this paper?
Where does it eventually fall down? Show where it stops working.
What assumptions have I made? i.e. data is sync'd at t=0.
Can it do more or less, better or worse than DTW?
Parameters and what they affect:

- window size
- step size
- length of signal
- noise of signal
- periodicity of signal

4.1 SPHERE Data Tests

Evaluation of SPHERE tests here.

4.2 Results

Results here.

Chapter 5

Conclusion

Future work: Use these assessments as a means of rejecting false positive skeletons. Skeletons are the format in which the data comes from the cameras. It contains information on the location of the main joints of the person i.e. elbow, wrist, knee etc.

Bibliography

[1] SPHERE. Sphere website, 2016. [Accessed 2nd April 2016].