Synchronizing Time-Drifted Accelerometer Data Using RGBD Cameras

Progress Report 30/03/2016

Project Outline

Within the SPHERE research group, there is a wearable accelerometer which will induce a drift between the true time and the local time on the device. Currently it is unknown what the true drifts are, but they are enough that the wearable needs to be re-synchronised for every use. The accelerometer is used for 30 minutes at a time, whilst someone follows a script around the SPHERE house. By considering video data from RGBD cameras around the house, which provides data on where the person is, together with the acceleration data from the wearable it should be possible to automatically correct this drift so that the accelerometer and the video remain synchronised over whole sequences.

The project could be broken down in the following steps:

1. Given a perfectly synchronised set of data from the accelerometer and the cameras, introduce artificial drifts between data sets and determine whether compensations can be made.
2. Apply whatever algorithm is used in step 1 to real drifted data already collected in the SPHERE house in three locations; hallway, living room and kitchen. Each may elicit different reliabilities for re-synchronisation.
3. 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.

Implicit to all of these steps will be a fair amount of data formatting and visualisation. At first glance, it seems that the bulk of this project will be looking to create an algorithm which will synchronise the two data sets. I will be using Python for a the implementation and I may also find SciPy useful which is an open source library of scientific tools for Python.

Progress

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:

- 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. I came across a few papers which 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. One obstacle is that the video data is extremely noisy and differentiating it results in amplified noise. I need to look in to solving this problem either with data smoothing or tackling the problem without differentiation.

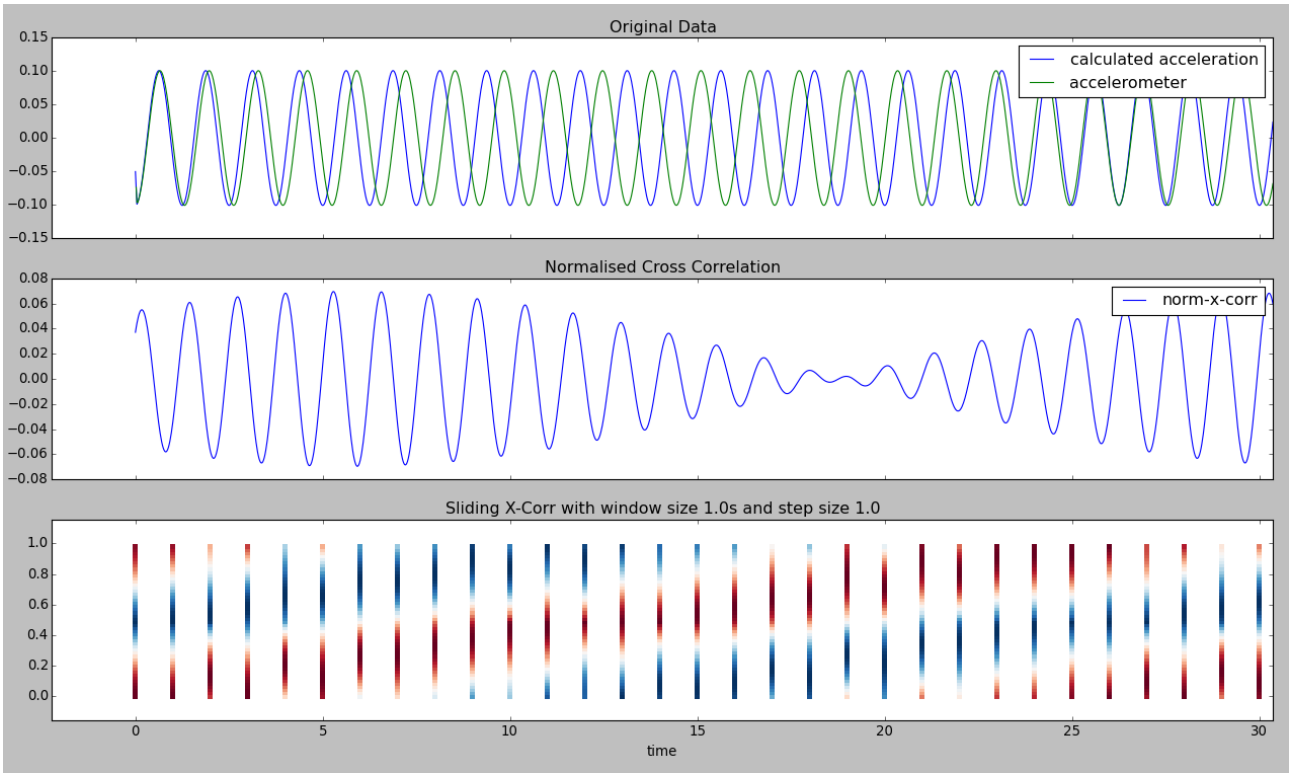
The data from the RGBD cameras also has a huge feature set after the data formatting and pre-processing where I calculated velocity and acceleration. There are 10 joints which may all be correlated in some way to the acceleration of the right wrist, and each joint has 3 dimensions. There is a decision to be

made to choose which one(s) to use?

Another complication is that some research papers claim that PCA to reduce dimensionality is useful when compensating for the accelerometer rotating along 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.

From here, I plotted the cross correlation of this data as before 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.



I have also began to research Dynamic Time Warping and how that could help find a solution to the problem.

Still to do:

1. Sliding window cross correlation vs DTW?

2. Track and fix drift for simple periodic data.
3. Track and fix drift for simple noisy periodic data.
4. Track and fix drift for complex noisy periodic data.
5. Non-periodic data.
6. Use this algorithm on real SPHERE data:
    1. How noisy is the video data?
    2. Could different joints be less noisy but still correlated (shoulder vs wrist)?
    3. Can we avoid differentiating or integrating?
    4. Which components of joints should we use? X, Y, Z? Do we need to use PCA?