Implementing Dual Foot Mounted Configuration with Oblu

July 2018

Shreeshail Hingane(160294)

Rohin Garg(160583)

Shivank Garg(160658)

Group 10, CS664

Mentored by-

Professor Amey Karkare

Amit Gupta Sir(Founder, GT Silicon)

Acknowledgement:

We express our gratitude to Prof Amey Karkare and Amit Gupta Sir for providing us the opportunity to work on the project “On Oblu and Cloud” as part of this course. We express our gratitude to GT Silicon and its employees for mentoring us in different aspects of IoT. Without their moral and technical support, we would not have been able to complete this effortful task. Overall, we thank them for providing us with the opportunity to make something creative along with gaining.

Overview

Oblu is motion sensing development board. It can be mounted on a shoe to give the PDR data. There is a systematic heading drift error which grows with time. To reduce this error, we follow an intuitive approach of fusing data from two oblu devices mounted on left and right foot, and correcting their data to reduce drift error.

Goals

1. Sending data to cloud (firebase in our case).
2. Fusion of data from Left and Right Foot mounted Oblu devices to get correct data.
3. Displaying Raw Left and Right Coordinates and Corrected Left and Right Coordinates on a Graph.
4. Benchmarking the Performance of dual foot-mounted Oblu against single foot mounted configuration. Measuring Distance Error, Drift Error and height

Hardware and Tools Required-

- Oblu motion sensor equipped with onboard IMU array, floating point processor, ESP and BLE modules.
- Arduino(to program the on-board ESP module).
- FireBase(a mobile and web application development platform).
- Amazon AWS servers(to collect data and run the fusion algorithm).
- Atmel Studio(to bootload the calibration files onto oblu)
- Arduino IDE
- USB cable, breadboard and jumper wires.

Libraries and Modules-

- Python2-pyqt4, tk and Matplotlib
- ArduinoJSON and firebasearduino
- Pyrebase, etc,

Specifications

1. Fusion Algorithm

- Our fusion algorithm works by identifying and correcting the erroneous data points that subsequently lead to drift error. Every time the cloud gets the coordinates of a new step, it is compared with a reference (for *right_step_data*, the reference is *previous_left_step_data* and for *left_step_data,* the reference is *previous_right_step_data*). If the distance between current step and previous step is more than a threshold, we reset the coordinates of current_step. We bring it within the threshold distance of the previous step while keeping it along the same line joining the original current_step and previous steps.
- **Note:** It is important to note that before we apply the previous algorithm, the coordinates must be adjusted so that they are in the same co-ordinate system. Hence the legs must be in a fixed position beforehand. This way there will a specific(and fixed) bias separating the co-ordinates and this can be taken into account while making corrections.
- **Drawbacks -** One major problem arises when oblu loses connection temporarily. Thus you lose a few(2-3 readings) . The next reading that is received is naturally far away from the previous steps and even though this data might be accurate, it is perceived as being wrong and is thus corrected for. This induces a constant error in reading that propagates into further measurements.

2. Firebase

- Sent data to firebase using ESP8266. Oblu sends a packed byte array to ESP8266 which we have to unpack before sending it to firebase, otherwise Firebase will show absurd data.
- Currently the firebase is able to receive successfully initially few readings but gives “header parsing” due to large delay because of updating the coordinates.
- So we have implemented to send a JSON format all at once to reduce the number of times delay occur.
-

3. Raw and Corrected data plot

- Used the matplotlib animation library to create real-time plots.

- Matplotlib is taking input from ".txt" files which are also updating in real-time with every new entry from Oblu.
- Left and Right oblu graphs are given in 2 subplots with different shades for raw and corrected reading.
- Animated module of matplotlib is important feature for real time plotting

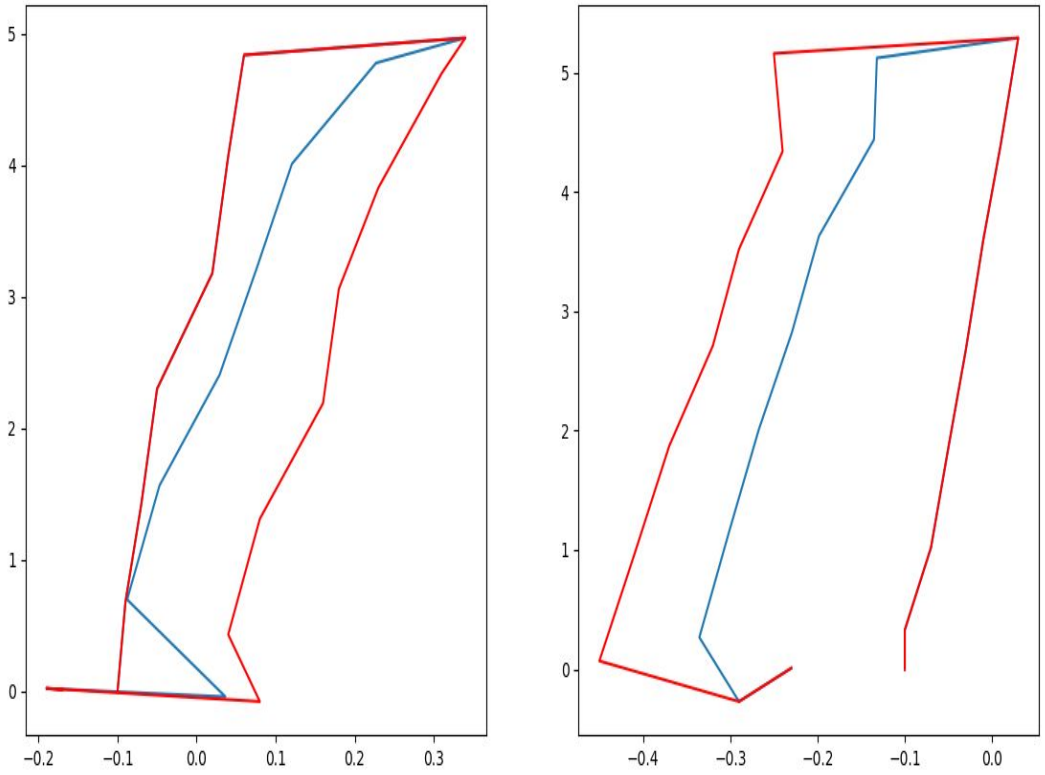
4. Amazon Cloud

- Setted up an Amazon EC2 instance for computation of data from firebase database.
- Accessing the VM instance using openssh and filezilla.
-

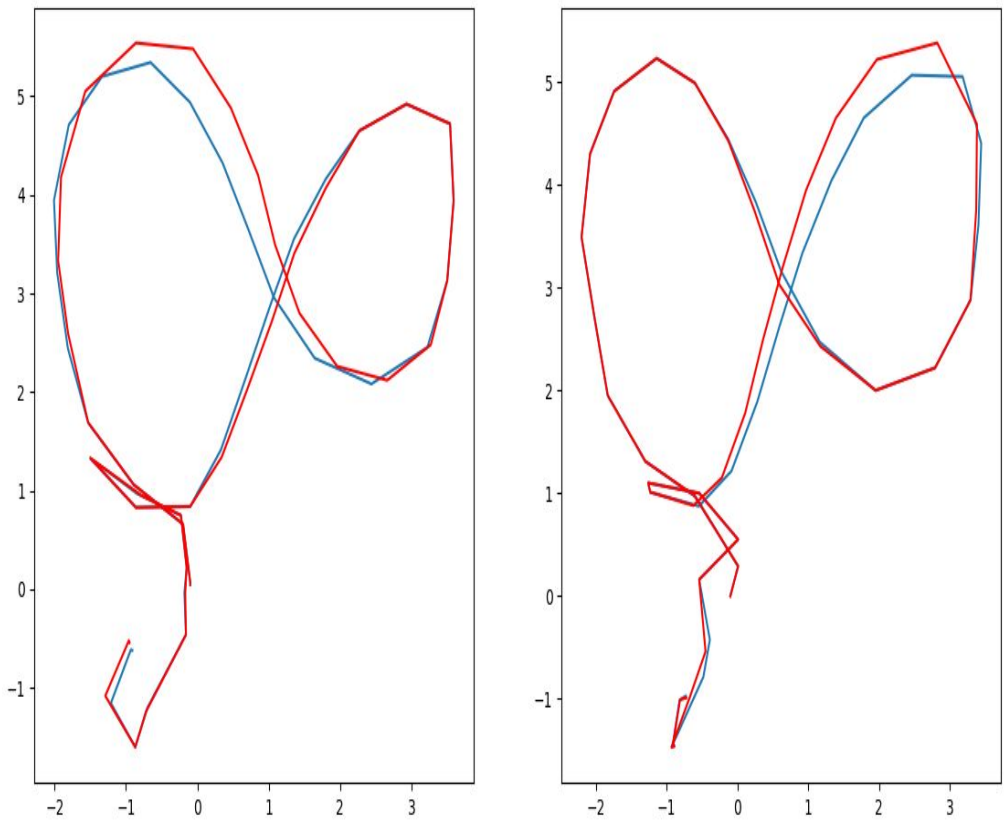
5. Benchmarking

- Checking the drift and distance error by walking along the fixed path many times and then returning to same point to check the drift from initial zero value.

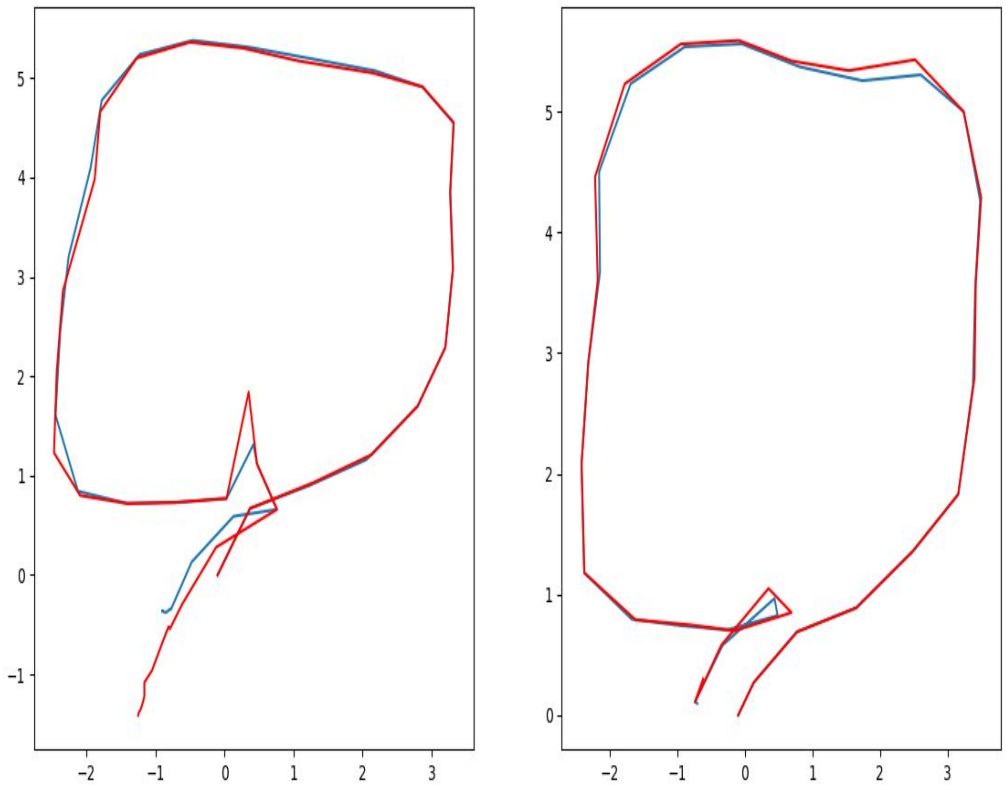
Experimental Results:



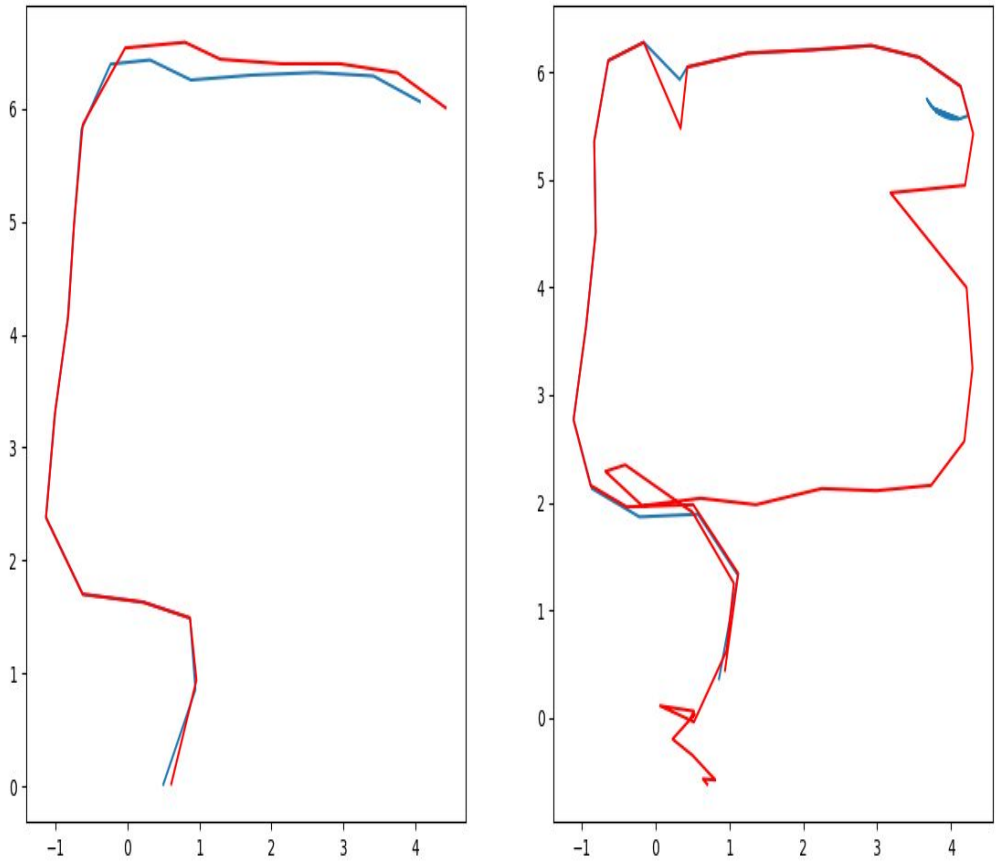
(Red: Original, Blue: Corrected): This clearly demonstrates the effect of the algorithm. The actual path is nearly straight line travelled back and forth. A sharp and sudden turn at one end is enough to offset the device into making an erroneous measurement and error is carried throughout. The fusion algorithm however, recognizes this disparity(large increase in distance between current and previous step) and corrects it immediately.



(Red: Original, Blue: Corrected): Another example of path correction. This time however, the difference is small and not as stark as the previous case, primarily because there are no sharp turns in shape of 8 to offset the values too much.



(Red: Original, Blue: Corrected): A clear demonstration of the working principle of the fusion algorithm. The sudden changes are scaled down to the length of the threshold. This is clearly seen in the bottom as the kink is scaled down, giving a much smoother path.



(Red: Original, Blue: Corrected): One of the drawbacks of the current fusion algorithm. In the right side of the figure, see how the initial kink(on the top) is corrected for properly. Next: The left oblu disconnects, thus the left graph stops. Now notice the second kink on the right; note that since there is no accurate reference now to correct(the left step data has stopped arriving), the correction(the blue line coming inward) is weird and not at all the result we expected.

Problems Facing and Proposed Solution-

1. Sending data to firebase requires some time which delay the data synchronization b/w oblu and Laptop. This gives a error related to header parsing when ObluIoT is runned. If Acknowledgement is also implemented then delayed will be further increased due to receiving it to ESP8266

   Solution - A Single json string can be updated at once, instead of updating it separately.

2. Both Oblu give data reading in their relative frame. So, it is not possible to find the distance b/w the two oblu without finding the relation b/w the two frame.

   Solution - A simple implementation of choosing them to start from given position will help to know initial relation b/w two frames. But it is not a good implementation as there will misalignment of axis and distance between Shoe Mounted Oblu. A more foolproof method is required for fusion algorithm

3. Problem with Soldering and loose connection - ESP8266 wiring gets desoldered frequently due to changing of circuit from oblu and breadboard. Also the procedure of programming ESP8266 is cliche and lengthy, which makes it difficult to debug the issues.
4. Currently, we are able to successfully retrieve data from firebase on Amazon EC2 instance using stream feature of pyrebase, but due to lack of time, we are not able to use JSON data format output to do fusion computation and plotting graph. If we had little more time, then we had done it on EC2 instance.

Future Prospects

- We could have a more sophisticated fusion algorithm that also takes into the account some of the problems we have mentioned above.

- Since the points are plotted with respect to the oblu's initial orientation, optimal results can be obtained only when the oblu's are oriented in the exact same way before starting. In future there can be some additional functionality joining the two oblu devices which can determine their relative orientations in the beginning, thus eliminating the need for the user to have to take care of the initial orientation.

- A highly complex fusion algorithm, some more computation power and more hardware resources could lead to eventual applications where moving entities need to know each other's location and communicate. Such present day applications include the robots that Amazon uses in their factories and warehouses, and also swarms of drones or miniature bots.

References

1. https://drive.google.com/open?id=1T-gRIpCPfAYkC60Hd0flxpZ9Efm6oVto
2. https://drive.google.com/file/d/1c04CWGTAfUzHT0wCtyeseT8j0tvQoeKa/view
3. https://www.slideshare.net/gt-silicon/foot-mounted-pedestrian-navigation-systems
4. https://www.slideshare.net/gt-silicon/longterm-performance-evaluation-of-a-footmounted-pedestrian-navigation-device
5. https://www.tindie.com/products/obluIoT/oblu-a-shoe-mounted-indoor-gps/
6. https://firebase.google.com/docs/?authuser=0
7. https://github.com/thisbejim/Pyrebase
8. https://aws.amazon.com/documentation/
9. https://github.com/bblanchon/ArduinoJson
10. https://arduinojson.org/v5/assistant/