Adjustment Of Google Maps Time For Personal Speeds Using And IMU and GPS

Benny Hansel
Northeastern University
hansel.b@northeastern.edu

Azizollah Taheri
Northeastern University
taheri.a@northeastern.edu

Shubhangi Pulipati
Northeastern University
pulipati.s@northeastern.edu

Anvitha Anchala
Northeastern University
anchala.a@northeastern.edu

***Abstract*—The widespread use of GPS technology in navigation systems has made it easier for people to navigate unfamiliar environments. However, GPS-based systems do not take into account the different walking speeds of individuals, which can lead to inaccurate predictions of travel time. In this paper, we propose a method for adjusting Google Maps estimated travel time by incorporating data from an inertial measurement unit (IMU) and GPS. Our approach uses the IMU to measure the user's walking speed and adjust the estimated travel time accordingly. We present experimental results that demonstrate the effectiveness of our method in improving the accuracy of estimated travel time, and we discuss potential applications of this technology in real-world scenarios, such as emergency response and transportation planning.**

***Index Terms*—GPS, IMU, navigation**

I. Introduction

Google Maps has become a ubiquitous tool for navigating through unfamiliar environments. It provides estimated travel time for various modes of transportation including walking, biking, and driving. However, the estimated travel time is based on an average walking speed which according to Google Maps is 1.3 m/s. It does not account for individual variations in walking speed. This limitation can lead to inaccurate predictions of travel time and can be frustrating for users who are in a hurry or have mobility impairments.

To address this limitation, we propose a method for adjusting Google Maps estimated travel time by incorporating data from an inertial measurement unit (IMU) and GPS. An IMU is a sensor that can detect changes in acceleration and orientation, which can be used to estimate the user's walking speed. By combining this information with GPS data, we can accurately measure the user's movement and adjust the estimated travel time accordingly. Our project is divided into two levels .The first level used only IMU to calculate the velocity and then the distance .The second level uses IMU to calculate the velocity for the scaling factor and the GPS to obtain the distance .For both the levels , we obtained the Google maps time and distance using a web scraping tool which we incorporated in the driver code.

Our approach has several advantages. First, it is more accurate than the current method used by Google Maps, which only considers the average walking speed. Second, it is adaptable to individual variations in walking speed, which can improve the user experience for people who walk at a slower or faster pace than the average. Third, it can be used in real-time to adjust the estimated travel time based on the user's current speed, which can be useful for emergency response situations or transportation planning.

To validate our approach, we conducted experiments by walking with a bicycle along a predetermined path while carrying an IMU and a GPS receiver strapped to the front of the bike. We compared the estimated travel time using our method to the actual travel time and found that our method significantly improved the accuracy of estimated travel time compared to Google Maps

II. Related Work

Previous work in pedestrian navigation systems has focused on using smartphone sensors, such as the inertial measurement unit (IMU) and GPS, to estimate walking speed and adjust estimated travel time. For instance, Altini et al. (2012) [1] proposed a self-calibration method for walking speed estimations using smartphone sensors. Itini et al. (2015) [2] proposed an adaptive pedestrian navigation system that uses a smartphone with an integrated IMU and GPS to estimate the user's walking speed and adjust the estimated travel time accordingly. Other studies, such as [3][4]and [5], have investigated personalized methods for estimating walking speed using smartphone sensors. Jimenez [6] conducted a comparative analysis of pedestrian dead-reckoning (PDR) algorithms using low-cost microelectromechanical systems (MEMS) and IMU sensors in contrast with other algorithms.

However, these approaches primarily focus on improving accuracy in real-time navigation using smartphone sensors [7], and they do not address the problem of inaccurate estimated travel time in map applications like Google Maps. In contrast, our proposed method aims to adjust the estimated travel time in map applications by incorporating data from IMU and GPS sensors. Moreover, our method is adaptable to individual variations in walking speed and can be used in real-time to adjust the estimated travel time based on the user's current speed. Therefore, our proposed method addresses a different problem and provides a new solution for improving the accuracy of estimated travel time in map applications.

III. Set up

Materials used: Bicycle with a secure front wheel and a flat surface on the handlebars or frame for mounting sensors,

Inertial Measurement Unit (IMU) sensor, Global Positioning System (GPS) sensor, Tape for mounting sensors to the bike, Computer for collecting and analyzing data.

Initially, we planned to attach the sensors to the ankle or knee of the participant, but we found that the sensors were too sensitive to movements. Therefore, we decided to use a bicycle instead and strap the sensors to the front of the cycle.

To set up the experiment, we first prepared the bicycle by ensuring that it was in good working order and had a stable front wheel. We then attached the IMU sensor to the bike in a way that allowed it to measure movement in the X, Y, and Z directions. We oriented the sensor so that the X direction faced forward. We also strapped the GPS sensor to the front of the bike in a location that provided an unobstructed view of the sky to ensure good signal reception.

Using straps or other secure attachment mechanisms, we ensured that both sensors were firmly attached to the bike and would not move during the experiment. We then walked the bike at a steady speed of 1 to 2 m/s for a predetermined distance (e.g., 100 meters) and collected data from both sensors using a mobile device or computer.

After collecting the data, we analyzed it to calculate walking time and distance, as well as any other relevant measures (such as speed or stride length). Throughout the experiment, we took care to ensure the accuracy and reliability of our results by calibrating the sensors and conducting multiple trials to confirm our findings.

In our report, we will describe the materials and methods we used in more detail, as well as any challenges we encountered and any limitations of our methodology. Overall, we believe that our approach using a bicycle and front-mounted sensors provided a more stable and accurate way to measure walking time and distance than our original plan to attach the sensors to the participant's body.

IV. Data Collection

A. LEVEL 1 Data Collection

The figures from 1-3 shows the level 1 data collection.

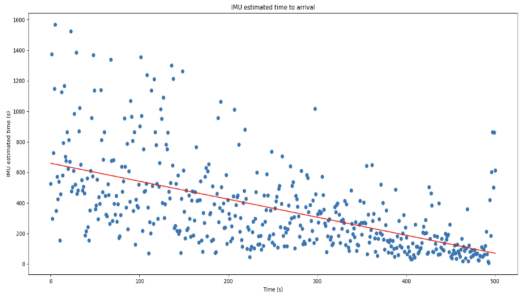

Fig. 1. Plot showing values of estimated time vs time of experiment

B. LEVEL 2 Data Collection

The figures from 4-6 shows the level 2 data collection.

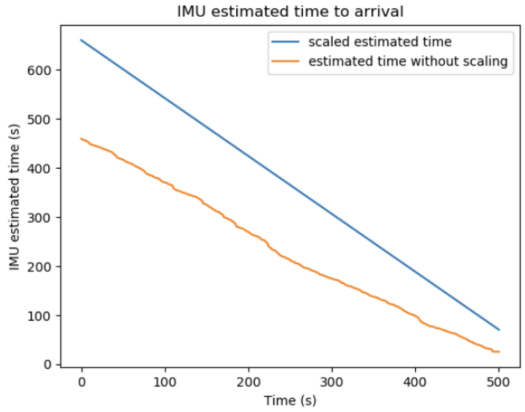

Fig. 2. Best fit line of estimated time compared to googles estimated time

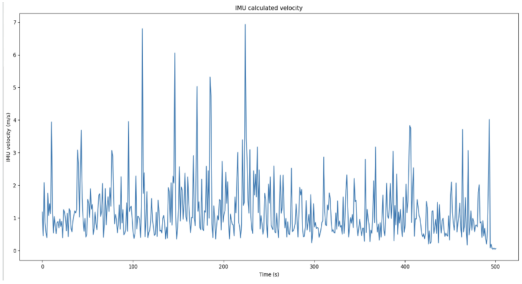

Fig. 3. Calculated velocity from IMU over time

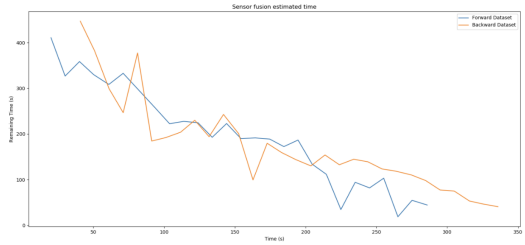

Fig. 4. Estimated time during route with forward and backward datasets

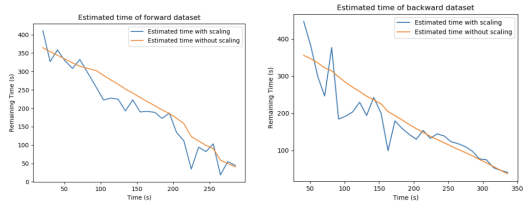

Fig. 5. Estimated times of dataset in comparison with google maps estimated time

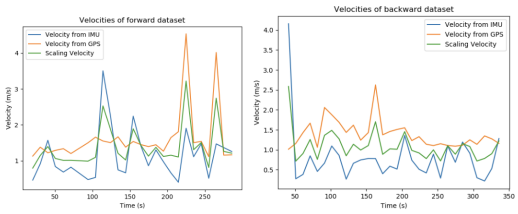

Fig. 6. Velocity calculation comparison of IMU, GPS, and simple sensor fusion

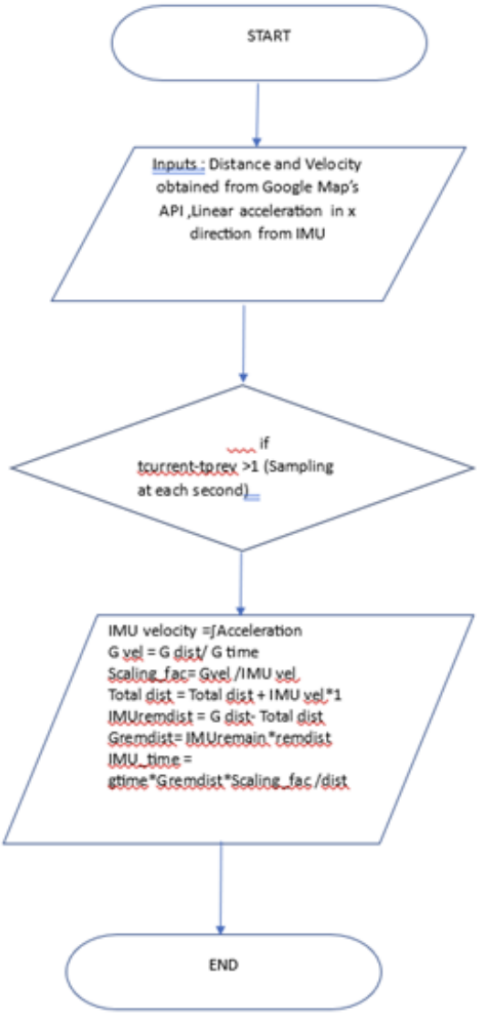

Fig. 7. LEVEL 1

V. Workflow

A. LEVEL 1

B. LEVEL 2

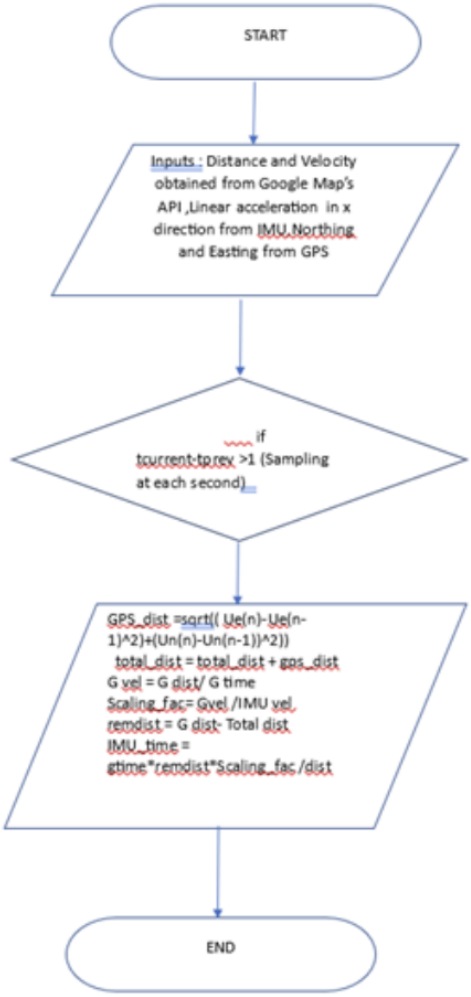

Fig. 8. LEVEL 2

VI. Analysis

This project evolved through our level 1 data collection. Figure 1 illustrates that there were large amounts of noise in the calculation of the estimated time, and Figure 3 shows that the velocity calculation from the imu was the source. The algorithm for the level 1 data collection had a filter and was integrating over at least 9 data points to find a single velocity value every 5 seconds, and still experienced this much noise. The trend of the estimated time still closely matched the estimated time based on google maps as shown in Figure 2, so while there was a lot of noise to clean up, the algorithm and concept was functional.

In order to reduce noise of the estimated time, the algorithm of the imu had to be tuned and sensor fusion between the GPS and IMU could be utilized. The velocity calculation was also

used to determine the total distance travelled, and a GPS puck would improve this calculation significantly as it excels in giving accurate location readings. This distance can now be used in combination with our sampling time to get another measure of average velocity along with the IMU.

After making these changes, level 2 data collection was conducted. The route was shorter this time and 2 datasets were collected, one walking forward to the destination and the other walking back from the destination. The backwards data set had a walking speed slower than the forwards dataset, to see how the algorithm would react.

The changes in the velocity calculation proved to be very effective. Figure 5 shows how each dataset followed the estimated time with much less noise than the level 1 data. Both datasets matched up with the estimated time from google quite well, with differences coming in the local speed causing the expected variation between datapoints.

The sensor fusion between the IMU and GPS proved to be a large help in reducing noise. The velocities of each dataset are displayed in figure 6, and both the IMU and GPS can be seen exhibiting errors. The sensor fusion, simply taking the average between the IMU and GPS velocity calculation, successfully lessened the impact of these errors and lead to a smoother estimated time calculation.

VII. Challenges Faced

There were many obstacles the group overcame throughout the project, and many can still be improved upon. The initial plan was to have a web scraper collect the data from google maps, but many problems arose from this. Google maps had popups that contain the data needed by the scraper making it a lot more difficult, and all the data is rounded to tenths of miles and minutes leading to less accurate calculations. The google distance matrix API was much friendlier to use and contained more accurate data, with the drawback being it cost money. This meant we had to shift from updating frequently with google maps to taking the total distance and time google estimates once.

The driver for the sensor fusion also proved to be challenging. Originally the sensor fusion was done with launching 2 separate drivers, but for this project they would have to communicate. This would take time and add complexity, so the group instead chose to use 1 driver with multiple serial ports. While this solved the complexity and improved execution time, this meant the sampling rates of the 2 sensors would interfere with each other and lose some data. With 2 drivers the sampling rates would return to normal, and a smaller overall sampling rate may have been utilized. Alternatively, python supports threading which could also run 2 drivers simultaneously.

There were a couple of sources of error during the data collection. The IMU noise came from small oscillations and bumps in the round causing large acceleration that had little impact. Similarly, the IMU was only using the acceleration in the x direction, so any misalignment would lose acceleration to the other axis. This is likely the reason for the IMU velocity being consistently lower than the GPS velocity in figure 6. The GPS velocity calculation also contained errors, as calculating the average velocity and distance between 2 points assumes a straight line. With larger sampling times turns and sharp angles have relatively lower distances and higher velocities.

VIII. Conclusion

Overall, the project was a success. The level 2 algorithm proved the concept of scaling the google maps time with local speed data, especially after the sensor fusion with the GPS and IMU. Carrying around a GPS and an IMU is inefficient, but Smartphones have a GPS and Accelerometer in them, so this algorithm could be implemented locally using them instead of the bulky setup of this project. The practical application and accuracy of this project was shown, and with more tweaking to the sensor fusion and filtering this algorithm will improve as well.

IX. Future Works

The future scope and improvement of this project includes the creation of GUI instead of displaying the time in the terminal window. We can create an application which uses the gps and imu present in the mobile phone to give the user a personalized estimate of remaining time. This can further be implemented in fitness watches which can help the user with tasks like training for a marathon or a race.

References

[1] Siavash Hosseinyalamdary, Yashar Balazadegan. Tracking 3D Moving Objects Based on GPS/IMU Navigation Solution, Laser Scanner Point Cloud and GIS Data. 2015

[2] Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA, 20–26 June 2005.

[3] Korn, M.; Holzkothen, M.; Paul, J. Color supported generalized-ICP. In Proceedings of the 2014 International Conference on Computer Vision Theory and Applications, Lisbon, Portugal, 5–8 January 2014.

[4] Johan Wahlstrom, Isaac Skog. Map-Aided Dead-Reckoning Using Only Measurements of Speed. IEEE Transaction on Intelligent Vehicles. January 2017.

[5] Held, D.; Levinson, J.; Thrun, S. Precision tracking with sparse 3D and dense color 2D data. In Proceedings of the 2013 International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013.

[6] Buehler, M.; Iagnemma, K.; Singh, S. The DARPA Urban Challenge: Autonomous Vehicles in City Traffic; Springer: Berlin, Germany, 2009.

[7] Viola, P.; Jones, M. Robust real-time object detection. Int. J. Comput. Vis. 2004, 57, 137–154.

[8] Papageorgiou, C., Oren, M., and Poggio, T. 1998. A general framework for object detection. In International Conference on Computer Vision.

[9] Schneiderman, H. and Kanade, T. 2000. A statistical method for 3D object detection applied to faces and cars. In International Conference on Computer Vision.

[10] P. Besl, N. McKay. "A Method for Registration of 3-D Shapes," IEEE Trans. on Pattern Analysis and Machine Intel., vol. 14, no. 2, pp. 239-256, 1992.

[11] D. Hahnel, W. Burgard, S. Thrun. "Learning compact 3D models of indoor and outdoor environments with a mobile robot," Robotics and Autonomous Systems, vol. 44, pp. 15-27, 2003.

[12] Y. Chen, G. Medioni. "Object Modeling by Registration of Multiple Range Images," Proc. of the 1992 IEEE Intl. Conf. on Robotics and Automation, pp. 2724-2729, 1991.