Ubiquitous Computing

Final Project

Kraunelis, Ravi, Fitzpatrick

**Problem Statement**

The problem we aim to address is losing GPS signal while using navigation apps on mobile devices. According to a Pew study performed in 2015 [http://www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/], 67% of users use their smartphones for turn-by-turn directions. A GPS receiver needs to communicate with 4 satellites in order to fix a location. While losing GPS signal can occur under a number of circumstances, the most common scenario is when the line of sight is obstructed by the environment. This includes natural formations such as a mountains and dense forests, as well as manmade structures such as bridges, buildings and tunnels. Losing GPS signal poses many significant problems for drivers. Perhaps the most significant of these is distracted driving, caused by diversion of attention to the device when signal is lost. Secondly, there is an environmental impact caused by wasted fuel if the driver gets lost or has to diverge from the optimal route. Finally, losing GPS signal can have economic impacts, such as missing a flight, being late for work, or delaying a delivery.

**Related Work**

Much of the work in the area of dead reckoning via smartphone sensors is centered around tracking location of pedestrians. A major motivation for this is to track user location in so-called urban canyon environments. These are areas where GPS signal is degraded due to human built structures, such as city streets or inside buildings.

Kakiuchi, et al. describe a pedestrian dead-reckoning technique which uses a smartphone placed in the user’s pocket. Their model uses the accelerometer and magnetomer sensors to determine the orientation of the device, then detect steps and direction of travel. Additionally, the authors provide a model for stride length estimation, which can adjust the localization according to whether the user is walking or running. [3]

A few papers exist on the topic of dead reckoning via smartphones for vehicle-related applications. Unlike our goal of providing a general purpose, outdoor navigation when GPS signal is lost, the approaches taken in these papers focus on estimating location for other purposes, such as mapping.

Bojja, et al. use vehicle odometer and smartphone gyroscope information to provide indoor localization. The odometer information is read from the OBD-II port and transmitted via Bluetooth to the smartphone. They use the famous particle filtering algorithm to determine the current position based on sensors. Their approach provides accurate localization, however it depends on a 3D model of the building, which may not be available in all circumstances. [1]

Gao, et al. use dead reckoning to crowd source a road map from user’s smartphones inside vehicles. They rely only on the smartphone’s accelerometer to detect bumps and turns, and perform a double integration on device acceleration to discover which direction the vehicle is headed in. This method is sensitive to smartphone placement, which may not be ideal for users. [2]

The most relevant work we found was Walter, et al. Their approach is similar to Bojja, et al., where the vehicle information is read from the OBD-II port and the smartphone sensor used is the gyroscope. For vehicle sensor information, the speedometer is used instead of the odometer. They use a modified Kalman filter to predict the current location based on the previous states. They perform test drives and simulations to prove the accuracy of their system, and show that using the sensors fusion can improve location accuracy with no GPS signal over the baseline. [4]

**References**

[1] Bojja, Jayaprasad, et al. "Indoor localization methods using dead reckoning and 3D map matching." *Journal of Signal Processing Systems* 76.3 (2014): 301-312.

[2] Gao, Ruipeng, Guojie Luo, and Fan Ye. "VeMap: Indoor Road Map Construction via Smartphone-Based Vehicle Tracking." Global Communications Conference (GLOBECOM), 2016 IEEE. IEEE, 2016.

[3] Kakiuchi, Noriaki, and Shunsuke Kamijo. "Pedestrian dead reckoning for mobile phones through walking and running mode recognition." *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*. IEEE, 2013.

[4] Walter, Oliver, et al. "Smartphone-based sensor fusion for improved vehicular navigation." *Positioning Navigation and Communication (WPNC), 2013 10th Workshop on*. IEEE, 2013.

**Proposed Solution**

Our solution to this problem is to leverage the device’s sensors and vehicle speed to estimate the current position based on the last known fix, using a technique called dead reckoning. The solution contains both hardware and software components. The hardware component contains an OBD-II adapter ($45) and an Arduino 101 ($30).

<http://freematics.com/store/index.php?route=product/product&path=20&product_id=83>

<https://store.arduino.cc/usa/arduino-101>

The software component (Android only) involves implementing a custom LocationProvider class that will provide location updates via dead reckoning. There are advantages and disadvantages to this approach. The main advantage is that a LocationProvider can generate location updates for any application on the system. This means that users can continue to use their favorite navigation app, whether it is Google Maps Navigation, Waze, or Mapquest, and not have to install and learn to use another third party navigation app. The disadvantage to this approach is that only system apps are allowed to implement a LocationProvider, therefore we’d require a workaround, until this LocationProvider was accepted into the Android platform. The workarounds could include rooting the device to put our LocationProvider app on the system partition, building a custom ROM with our application installed, or using the MockLocationProvider with developer mode to get around the restriction.

The OBD-II hardware would be installed into the vehicle and used to provide vehicle speed updated to the LocationProvider app via Bluetooth. To calculate position, dead reckoning simply relies on speed multiplied by time. Using the speed reading together with the device’s magnetometer to get a bearing, we could calculate the current location and publish updates throughout the Android system to any app that was listening – including navigation apps.

Diagram of Proposed Solution:

Above is a diagram of our proposed solution. We intend to utilize the vehicles OBD-II adapter and Arduino 101. This hardware component will communicate with the user’s smartphone device (specifically the LocationProvider class) to provide dead reckoning updates (speed and time). The GPS receiver within in the smartphone will then communicate with GPS signals should GPS signal be available. Otherwise, our dead reckoning implementation will continue to provide turn-by-turn updates.

[1] Woodward, Collin. “This BMW Concept has No Doors, No Roof, and a Huge Gesture Controlled Screen.” Road & Track Magazine. (2015)

[2] Huang, Stanley. “My home-brew Arduino OBD-II connection kit. Arduino Development Forum. (2012).

[3] “LG&E and KU go live with iFactor’s smartphone application.” Kubra. (2011)

[4] Teletrac Navman Solutions.

**Evaluation Methodology**

The purpose of this evaluation is to investigate whether location fixed by dead reckoning can reliably be used with navigation apps when GPS signal is lost. By nature, dead reckoning relies on previous position to estimate the current position, therefore any errors in location estimation are cumulative. To ensure that we don’t end up guiding the user off course, we must minimize error at any given position fix. To test this, a device using our LocationProvider will be used to navigate to a number of destinations. The driver should be instructed to follow the route guidance, so long as it is safe to do so. We can simulate loss of GPS signal simply by turning the GPS radio off on the device. For each destination, we should repeat the experiment with route guidance guided by GPS. We can then compare the dead reckoning results to the baseline GPS route based on time to destination, whether the driver arrived at the correct destination, number of wrong turns, etc.