Maturaarbeit

Entwicklung eines Arduino-basierendem Embedded-Systems zur Höhenmessung

Unter der Verwendung von Sensor Fusion Algorithmen

Riccardo Orion Feingold

Betreut durch

Stefan Rothe

30. Julie 2019



Gymnasium Kirchenfeld

Abteilung MN

Inhaltsverzeichnis

**Es wurden keine Einträge für das Inhaltsverzeichnis gefunden.**

Vorwort

Abstrakt

﻿In this paper a location estimation algorithm is briefly presented - especially focused on the vertical position. The idea is based on the combination of a Map-Matching algorithm and a Kalman Filter. Both are implemented in a altimeter, which consists out of a GPS shield and a sensor. While the GPS is giving the x,y and z coordinates as well as the velocity, the sensor can measure the temperature, pressure and the altitude over mean sea level (MSL) by using the so called international height formulae (or another one). The goal of this paper is to introduce some basic knowledge and to give a short answer on to the question: How can the vehicle’s positioning be improved?

Einleitung

﻿In den letzten Jahren hat die Positionsbestimmung an Wichtigkeit zugenommen, da immer mehr Verkehrsmittel genaue Positionsdaten benötigen. Sie erlauben nicht nur eine Standort-Abfrage, sondern aus ihnen lassen sich auch Ankunftszeiten an einem gewissen Ort im Voraus bestimmen oder u.a. verlorene Smartphones wiederfinden. Doch sie werden in der Zukunft weiter an Wichtigkeit zunehmen. Zum Beispiel wird sie für selbstfahrende bzw. selbstfliegende Verkehrsmittel von enormer Bedeutung sein, vor allem für das letztgenannte. Denn hier müssen drei Koordinaten genau sein: Einerseits die X und Y Koordinaten, und die Höhe. Letzteres wirft die häufigsten Probleme auf, denn da treten die meisten Abweichungen in der Messung auf, zum Beispiel beim GPS (Globale Positioning System) oder Altimeter (Barometer, das die Höhe mit Hilfe einer Höhenformel berechnet) - die beiden einfachsten und häufig genutzten Varianten. Kombiniert man beide vernünftig, so dass sie sich gegenseitig unterstützen, lässt sich ein genaueres Höhenmessgerät entwickeln.

So ist das Ziel der Arbeit ein Embedded-System zur Höhenmessung zu entwickeln, welches nahezu in jeder Situation, bei denen GPS und Altimeter ungenaue Resultate liefern würden, zuverlässige Messungen garantiert. Dabei wird nicht nur ein Hardware Konzept geschaffen, sondern auch eine dazu passende Software geschrieben. Ebenfalls Gegenstand der Arbeit ist das Testen unterschiedlicher Arten der Höhenmessung auf ihre Stärken und Schwächen.

Um dieses Embedded-System bestehend aus einem GPS-Modul, einem Barometer und einem Beschleunigungssensor zu entwickeln, beruft sich der Autor auf ein Vorgehen, welches man unter den Namen “Sensor Fusion” kennt. Darunter versteht man das Kombinieren mehrerer Sensoren, mit dem Ziel bessere Resultate zu erhalten als mit einem einzelnen Sensor. Dieses Prinzip wird mit Hilfe zweier Methoden umgesetzt. Bei einer handelt es sich um das sogenannte Map-Matching (siehe Kapitel 3.1) und bei der anderen um den Kalman-Filter (siehe Kapitel 3.2). In Kapitel 4 werden die einzelnen Schritte im Rahmen einer Prozessdokumentation erläutert.

Zunächst werden einige theoretische Grundlagen zur Höhenmessung bekannt gegeben, die für das Verständnis der kommenden Probleme wichtig sind - ein solches Problem wäre z.B.: Warum gibt das GPS unpräzise Höhendaten in einem Parkhaus? Nebst dem GPS werden auch der Altimeter und der Beschleunigungssensor als weitere Varianten zur Höhenmessung erklärt. Danach werden die zwei Methoden - Map-Matching und Kalman-Filter - erläutert, sodass die benötigten Grundlagen für das nächste Kapitel gelegt sind. Dieses wäre die Prozessdokumentation des Produktes. Im Anschluss wird der Autor kurz auf die Resultate der Messungen eingehen und im Diskussionsteil interpretieren. Sprich, erklären warum die Resultate so aussehen und welche Bestandteile noch fehlen, damit es zu einem besseren Ergebnis kommen könnte.

Arduino vs. Raspberry Pi

//Programmiersprachen auch erwähnen

Arduino Mega 2560

Altimeter

# Höhenformel

# Herleitung «Internationale Höhenformel»

# DWD

# Unterschied

# Funktionsprinzip eines einfachen Barometers

BME280 Sensor

GPS

# Wie funktioniert die Positionsbestimmung?

# Effizienz der vertikalen Positionsbestimmung

# NMEA Sentences

Adafruit GPS Sensor //SD Card logging

Position berechnen mit Hilfe der Kinematik und dem Gravitationsgesetzt

# Inertial Measurement Unit Sensor

## Beschleunigungssensor

## Gyroscope

## Magnetometer

# BNO055

# Vergleich

User Interface mit LCD und Keypad

# Bedienkonzept

# LCD

# Keypad

Sensor Fusion

# Definition Sensor Fusion

Sensor Fusion ist ein Prinzip, das heute praktisch nicht mehr wegzudenken ist. Warum? Weil das was Forscher und andere messen, nicht nur ein Parameter ist, sondern eine ganze Palette. Es reicht nicht, nur die Temperatur zu bestimmen, wenn jemand Aussagen machen möchte über das Wetter von heute und morgen. Dazu braucht es noch einen Barometer, einen Windstärke-Messgerät und vieles mehr – sogar Satellitenbilder können auch behilflich sein. Also der Grundsatz für diese Kapitel lautet: Je mehr Informationen für ein System bekannt oder vorhanden sind, desto mehr kann darüber ausgesagt werden.

Aber bei Sensor Fusion geht es nicht nur um Aussagen, sondern auch darum, dass bessere Resultate erzielt werden können. Dies lässt sich durch Kombinieren der Sensoren bewerkstelligen. Die Idee dahinter sei, dass die Kombination bessere Messungen liefere im Vergleich zu einem einzelnen Sensor [3]. Also, statt nur den Barometer für die Höhenmessung zu verwenden, warum nicht gleich auch noch das GPS hinzunehmen!

Wie gut dann die Ergebnisse sein werden, hängt einzig und allein davon ab wie die Sensoren aneinandergefügt (im übertragenden Sinn) wurden, also welcher Algorithmus bzw. Filter verwendet wurde.

# Map-Matching

Map-Matching ist – wie bereits der Name schon sagt – ein Kartenabgleich. D.h. besitzt man einen Datensatz Z bestehend aus z.B. GPS-Messungen, so wird dieser mit einem Datensatz M, welcher Ortsinformationen von einer Karte besitzt, abgeglichen. Dabei beinhaltet M Datenpunkte, die als Referenzen verwendet werden – also invariant sind.

Diese Methode wird häufig im Bereich der Navigation angewendet. Dort tritt oft das Problem auf, dass die gemessenen GPS Koordinaten mit einer Ungenauigkeit (siehe Kap. Effizient der vertikalen Positionsbestimmung) verbunden sind und deshalb von den Ortsinformationen der Karte abweichen. Durch das Abgleichen der Messungen mit der Karte kann eine Kalibrierung der Sensoren bewerkstelligt werden [1].

Nun stellt sich die Frage wie man so ein Kartenabgleich implementiert. Eine Variante dies zu tun, ist via die Point-to-Point-Methode. Dabei handelt sich um eine durch den Autor abgeänderte Variante.

## Point-to-Point-Methode

Die Grund Idee der Point-to-Point-Methode ist eine Kalibrierung der Sensoren mit Hilfe von Fixpunkten – also Standorte bei denen sowohl Längen- und Breitengrad als auch die entsprechende Höhe angegeben sind. Dabei werden die Koordinaten vom GPS verglichen mit denjenigen vom Datensatz. Entspricht die Messung einem Datenpunkt, findet eine Kalibrierung statt – dazu später mehr.

Die Fixpunkte weisen - wie bereits erwähnt - eine relativ geringe Abweichung auf. Der Autor verwendet für sein Embedded-System die Daten von Swisstopo, die laut ihren Angaben eine Standardabweichung von +/- 0.5m aufweisen – dies gilt für alle drei Dimensionen [2]. Bei der Auswahl dieser Fixpunkte oder auch Referenzpunkte empfiehlt es sich, Standorte auszuwählen, die auf oder in der Nähe der Route ist. Ein Bespiel zeigt Abb. 1.



Abb. Foto von Autor: Referenzstrecke (rot) und mögliche Fixpunkte (grün)

Eine mögliche Implementierung eines Datensatzes aus Fixpunkten in C könnte wie folgt aussehen (aus Platzgründen wird nur ein Datensatz aus zwei Fixpunkten gezeigt):

#include <avr/pgmspace.h>

//1

typedef struct {

double lat;

double lon;

double h;

} X\_Y\_Height;

//2

const PROGMEM X\_Y\_Height data[] = {

46.90982,7.35846,585.7,

46.90934,7.35846,586.6

};

//3

double latitude = pgm\_read\_float\_far(&data[0].lat);

Als erstes wird ein neuer Datentyp für eine Struktur definiert, die drei Elemente besitzt: Latitude, Longitude und Höhe. Dann wird ein Array erstellt vom Typ X\_Y\_Height, welches in der ersten Kolonne die Latitude, in der zweiten die Longitude und in der dritten die Höhe beinhaltet. Doch damit diese Daten nicht den Arbeitsspeicher (den sog. SRAM) des Arduino Mega 2560 überfüllen, werden diese mit Hilfe von PROGMEM, das aus der «avr/pgmspace.h»-Bibliothek stammt, in den Flash-Speicher abgelegt. Punkt drei im Code zeigt lediglich wie man auf den Datensatz zugreifen kann.

Jetzt stellt sich natürlich die Frage wie und wann die Kalibrierung der Sensoren – d.h. in diesem Fall der Barometer und das GPS-Modul – mit Hilfe des Datensatzes stattfinden sollte.

## Kalibrierung des GPS Modul

Wie bereits erwähnt, werden die GPS-Koordinaten mit denjenigen vom Datensatz verglichen. Dabei wird nämlich für jeden Fixpunkt die Distanz zum Nutzer unter der Verwendung vom «Satz des Pythagoras» berechnet.

Damit das Programm die Werte besser abspeichern kann (als Vergleich: ein Meter entspricht in Grad 0.000008995), werden die Resultate in Meter umgewandelt, und es wird nicht die Wurzel gezogen, wenn der «Satz des Pythagoras» angewendet wird. Der Faktor wird wie folgt berechnet:

Faktor:

Während dem Prozess, bei welchem die unterschiedlichen Distanzen bestimmt werden, wird immer der am nächsten zum Nutzer liegenden Fixpunkt abgespeichert. Dann wird die Kalibrierung ausgeführt, falls der Abstand «Nutzer – Fixpunkt» kleiner ist als 16m. Ist dies der Fall berechnet das Programm einen additiven Kompensationswert, der dann den jeweiligen GPS-Koordinaten hinzugefügt wird. Dieser kann durch Subtrahieren des Fixpunktes von der Position des Nutzers berechnet werden.

## Kalibrierung des Barometers

Gleichzeitig kann auch der Barometer justiert werden. Wenn also der Nutzer sich in der gewünschten Nähe zu einem Fixpunkt befindet, so kann die Höhe beim Fixpunkt für die Berechnung des Druckes auf Meereshöhe genutzt werden (siehe Kapitel Höhenformel). Diese wird dann anstelle der 1013.15 hPa als neue Druckreferenz genommen.

Und so funktioniert die Map-Matching Implementation des Autors für die Kalibrierung des Barometers und des GPS. Als nächstes wird eine andere Anwendung gezeigt, die aus der Sicht des Autors als eine neue Variante, um die Höhe zu messen, dienen könnte.

## Idee von einem Höhenmodell

Diese Anwendung nutzt die Funktion des Map-Matching, um die drei nächsten Fixpunkte zu bestimmen. Da es sich hier jetzt um mehr als nur einen Punkt handelt, muss das Programm noch erweitert werden.

Wurden diese gefunden, wird mit Hilfe der Vektorgeometrie die Höhe beim Nutzer berechnet. Dies geschieht folgendermassen:

Als erstes wird aus den drei Fixpunkten ein Ebene in Parameter Form definiert. Dabei gilt für die Ebenen-Gleichung immer:

E: Ortsvektor + r\*Richtungsvektor\_1 + s\*Richtungsvektor\_2

Der Ortsvektor ist für diese Anwendung immer der am nächsten liegenden Fixpunkt. Dies ist deshalb der Fall, da der Autor eine Ebene aufspannen möchte, indem die Koordinaten des Nutzers Teil der Menge sind.

Um die Richtungsvektoren zu berechnen, werden die beiden anderen Fixpunkte vom Ortsvektor subtrahiert. Dabei muss beachtet werden, dass das Resultat nicht in die falsche Richtung zeigt.

## Probleme

# Kalman-Filter

## Definition Filter

## Gleichungen und Prozess

## Q und R: Was ist eine Covarianz Matrix?

### Einfluss von Q und R

## Kalman Gain

## Theoretische Überlegung für diese MA

## Was fehlt damit es funktionieren könnte?

//Überlegung einer App für das Smartphone

Bibliographie

1. Bevermeier, Maik et al.: *Barometric Height Estimation Combined with Map-Matching in a Loosely-Coupled Kalman-Filter* (2010) <https://www.researchgate.net/publication/224199908_Barometric_height_estimation_combined_with_map-matching_in_a_loosely-coupled_Kalman-filter>

(abgerufen am 25.03.19)

1. Swisstopo: swissALTI3D https://shop.swisstopo.admin.ch/de/products/height\_models/alti3D (abgerufen am 28.05.19)
2. Wikipedia: *Sensordatenfusion* (2019)

<https://de.wikipedia.org/wiki/Sensordatenfusion>

(abgerufen am 4.8.19)