VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

PROHLÍŽENÍ MAPY V MOBILNÍ APLIKACI POHYBEM ZAŘÍZENÍ

BROWSE THE MAP ON YOUR MOBILE DEVICE BY MOVING THE DEVICE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **DANIEL ANDRAŠKO**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VÍTĚZSLAV BERAN, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Andraško Daniel**
Program: Informační technologie
Název: **Prohlížení mapy v mobilní aplikaci pohybem zařízení**
Browse the Map on Your Mobile Device by Moving the Device
Kategorie: Uživatelská rozhraní

Zadání:

1. Seznamte se s problematikou vizuální odometrie s využitím jedné kamery a s možnostmi využití pohybu mobilního zařízení pro efektivní interakci.
2. Navrhněte způsob ovládání mobilních aplikací pracujících s mapovými podklady s využitím znalostí o pohybu mobilního zařízení v prostoru. Doplňte o další potřebné interaktivní prvky, aby práce s mapou byla co nejvíce intuitivní a efektivní.
3. Vytvořte demonstrační řešení s pomocí vlastní nebo existující mapové aplikace s využitím vhodných existujících nástrojů a knihoven.
4. Vyhodnoťte řešení na reálných úlohách a uživatelích.
5. Prezentujte klíčové vlastnosti řešení formou plakátu a krátkého videa.

Literatura:

- M. Sonka, V. Hlaváč, R. Boyle. *Image Processing, Analysis, and Machine Vision*, CL-Engineering, ISBN-13: 978-0495082521, 2007.
- Gary R. Bradski, Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*, ISBN 10: 0-596-51613-4, September 2008.
- Dále dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a částečně bod 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Beran Vítězslav, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 1. listopadu 2021

Abstrakt

Táto bakalárska práca sa zaoberá tvorbou mobilnej aplikácie, ktorá umožní jeho užívateľovi zobraziť mapu a pohybovať sa po nej, bez dotyku displeja, iba pomocou pohybu zariadenia. V teoretickej časti práce je bližšie popísaná inerciálna meracia jednotka a vizuálna odometria, ktoré je možné použiť pre zistenie pohybu zariadenia. Ďalej je popísaný Kalmanov filter, ktorý sa využíva pre spresnenie nameraných hodnôt senzoru akcelerometra. Ten je súčasťou inerciálnej meracej jednotky. V praktickej časti je popísaný návrh mobilnej aplikácie a implementácia daného návrhu. Záver práce obsahuje popis a vyhodnotenie testovania výslednej aplikácie na reálnych užívateľoch.

Abstract

This work deals with the creation of a mobile application that allows its user to view the map and move around it, without touching the screen, just by moving the device. The theoretical part of the thesis describes in more detail the inertial measurement unit and visual odometry, which can be used to determine the movement of the device. Next, a Kalman filter is described, which is used to refine the measured values of the accelerometer sensor. This sensor is part of an inertial measuring unit. The practical part of thesis describes the design of the mobile application and the implementation of the design. The end of thesis contains a description and evaluation of testing the final application on real users.

Klíčová slova

aplikácia, inerciálna meracia jednotka, Kalmanov filter, senzor, Android

Keywords

application, inercial measurement unit, Kalman filter, sensor, Android

Citace

ANDRAŠKO, Daniel. *Prohlížení mapy v mobilní aplikaci pohybem zařízení.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vítězslav Beran, Ph.D.

Prohlížení mapy v mobilní aplikaci pohybem zařízení

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vítězslava Berana Ph. D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Daniel Andraško
12. května 2022

Poděkování

Chcem sa poďakovať môjmu vedúcemu práce, pánovi Ing. Vítězslavovi Beranovi Ph.D., za odbornú pomoc a konzultáciu pri tvorbe tejto práce.

Obsah

Kapitola 1

Úvod

Používanie máp sa stalo bežnou súčast'ou života l'udí, či už z dôvodu cestovania, určovania najlepšej trasy alebo hl'adania miest záujmu vo svojom okolí. S vývojom technológií sa pomaly papierové mapy ubrali do úzadia a do popredia sa dostali mapové aplikácie na rôznych zariadeniach. Tie by mali poskytovat' jednoduchú a efektívnu možnost' ovládania mapy za účelom nachádzania určeného bodu záujmu.

Cieľom tejto práce je vytvorit' aplikáciu pre platformu Android, ktorá umožní jeho užívateľovi pohybovat' sa po mapovom podklade pomocou pohybu mobilného zariadenia. Mobilná aplikácia má zobrazovat' mapu a má obsahovat' interaktívne prvky, ktoré umožňujú jednoduchý a efektívny spôsob ovládania. Pre dosiahnutie požadovaných výsledkov je potrebné naštudovat' spôsob, akým sa zistí zmena pozície daného zariadenia. To umožňuje inerciálna meracia jednotka alebo vizuálna odometria, ktoré sú v práci bližšie popísané. K zisteniu pohybu mobilného zariadenia je použitý senzor akcelerometra, ktorý je súčast'ou inerciálnej meracej jednotky. Namerané hodnoty tohto senzoru niesu presné a preto je potrebné v návrhu mobilnej aplikácie, s tým počítat' a nájst' riešenie. Riešením môže byt' Kalmanov filter, ktorého úlohou je odstránenie šumu a spresnenie výsledkov merania určitej veličiny.

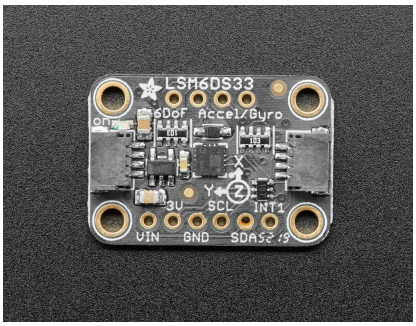
Súčast'ou práce má byt' testovanie a vyhodnotenie vytvorenej mobilnej aplikácie, na reálnych užívateľoch. Kľúčové vlastnosti mobilnej aplikácie majú byt' prezentované formou plagátu a krátkeho videa.

Kapitola 2

Inerciálna meracia jednotka

V tejto kapitole sa zaoberám jednotkou IMU, pretože je to zariadenie, ktoré je možné využiť pre zisk informácií o pohybe zariadenia. Pomocou nej dokážem zistiť polohu zariadenia a pri zmene tejto polohy aplikovať pohyb po mape.

IMU je elektronické zariadenie, ktoré slúži na meranie uhlovej rýchlosti, akcelerácie alebo orientácie telesa a následne získavanie týchto informácií. Toto zariadenie obsahuje akcelerometer, gyroskop a niektoré aj magnetometer. Väčšinou sa využívajú pri určovaní pozície a orientácie lietadiel ale taktiež lodí, raketoplánov, satelitov a iných zariadení. Existujú aj GPS zariadenia s implementáciou týchto senzorov. GPS zariadenia so zabudovanými senzormi dokážu pracovať aj v prípade, že GPS signály sú nedostupné, ako napríklad v tuneloch, v budovách alebo keď je signál rušený. Táto jednotka získava lineárne zrýchlenie tým, že využíva jeden alebo viac akcelerometrov. Podobne získava uhlovú rýchlosť využitím jedného alebo viac gyroskopov. Niektoré IMU obsahujú aj magnetometer, ktorý sa používa pre určenie smerovania. Typicky konfigurácia tejto jednotky pozostáva z jedného akcelerometra, gyroskopu a magnetometru na každú os z troch hlavných osí a to: sklon, natočenie a vybočenie [6] [15] [9].



Obrázek 2.1: IMU - Adafruit LSM6DS33 [1]

Veľkou nevýhodou akcelerometrov je to, že majú chyby merania. Aj malá chyba sa môže časom ukázať ako fatálna. Napríklad v navigačnom systéme kedy by aj malá odchýlka časom znamenala vysoký rozdiel od skutočnej hodnoty a hodnoty navigačného systému v ktorej by si myslel že sa nachádza. Existuje niekoľko definícií chýb, ktoré pri meraní akcelerometrom môžu nastať [6]:

[1] Obrázok prevzatí z: https://www.flickr.com/photos/adafruit/49390171508

- chyba odstupu (offset error) - stabilita výkonu (chyba merania počas nezmeneného stavu senzoru) a opakovateľnosť (chyba medzi dvoma meraniami v rovnakých podmienkach oddelená rôznymi podmienkami v čase medzi meraniami),
- chyba zarovnania (misalignment error) - pre nedokonalú mechanickú montáž senzorov,
- chyba vplyvu mier (scale factor error) - chyby citlivosti v dôsledku neopakovateľnosti a nelineárnosti,
- šum (noise) - závisí od požadovaného dynamického výkonu.

Existuje veľa rôznych IMU jednotiek, ktoré majú využitie v rozličných zariadeniach od chytrých hodiniek po nákladné lode a raketoplány. Tie sa delia podľa ceny a kvality snímania dát. Napríklad pre gyroskop a akcelerometer je výkon nasledovný [6]:

- od $0.1°/s$ do $0.001°/h$ pre gyroskop,
- od $100mg$ do $10\mu g$ pre akcelerometer.

To v reálnom prípade znamená, že jeden najmenej kvalitný akcelerometer ($100mg$) stratí svoju precíznosť na 50m za približne 10 sekúnd. Zatiaľ čo jeden najkvalitnejší akcelerometer ($10\mu g$) za približne 17 minút.

Senzory na platforme Android

Platforma Android poskytuje niekoľko senzorov, ktoré nám umožňujú sledovať pohyb zariadenia. Architektúra týchto senzorov sa líši podľa jeho typu a to na [4]:

- gravitačný, lineárno akceleračný, rotačný vektor, pohybový, počítanie krokov, meranie krokov sú senzory, ktorého základom je buď softvér alebo hardvér,
- akcelerometer a gyroskop sú senzory vždy založené na hardvéri.

Nasledujúca tabuľka 2.1 ukazuje senzory, ktoré je možné na platforme Android použiť. Je k ním pridaný popis, čo dané senzory merajú a jednotky, v ktorých sú namerané [4].

Senzor	**Popis**	**Jednotka merania**
TYPE_ACCELEROMETER	Zrýchlenie na osách x, y a z	m/s^2
TYPE_ACCELEROMETER _UNCALIBRATED	Merané zrýchlenie na osách x, y a z bez kompenzácie odchýlky	m/s^2
TYPE_GRAVITY	Gravitačná sila na osách x, y a z	m/s^2
TYPE_GYROSCOPE	Uhlová rýchlosť na osách x, y a z	rad/s
TYPE_GYROSCOPE_ UNCALIBRATED	Uhlová rýchlosť na osách x, y a z bez kompenzácie odchýlky	rad/s
TYPE_LINEAR_ACCELERATION	Zrýchlenie na osách x, y a z bez gravitačnej sily	m/s^2
TYPE_ROTATION_VECTOR	Zložka rotačného vektora na osách x, y a z	Žiadna
TYPE_SIGNIFICANT_MOTION	N/A	N/A
TYPE_STEP_COUNTER	Počet krokov od reštartu kým bol aktivovaný senzor	Kroky
TYPE_STEP_DETECTOR	N/A	N/A

Tabulka 2.1: Tabuľka senzorov na platforme Android

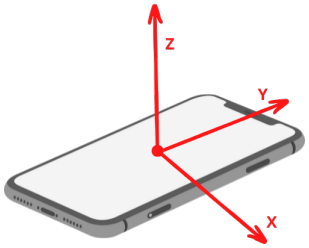
Dostupnosť senzorov na mobilných zariadeniach, sa líši podľa typu senzoru a platformy. V tabuľke 2.2 sú zobrazené senzory typu akcelerometer a lineárny akcelerometer a ich dostupnosť na zariadeniach s rôznou verziou systému Android. Pre zistenie zrýchlenia zariadenia budem využívať práve tieto dva senzory, ktoré bolo potrebné si naštudovať [4].

Typ senzoru | Platforma	Android 4.0	Android 2.3	Android 2.2	Android 1.5
TYPE_ACCELEROMETER	Yes	Yes	Yes	Yes
TYPE_LINEAR_ACCELERATION	Yes	Yes	n/a	n/a

Tabulka 2.2: Dostupnosť senzorov podľa platformy

Koordinačný systém

Koordinačný systém (obr. 2.2) na platforme Android pozostáva z troch hlavných os x, y a z. Väčšina senzorov meria oddelené hodnoty na každej osi zvlášť. Namerané hodnoty môžu byť kladné alebo záporné a to podľa smeru sily v danej osi. Napríklad pokiaľ sa mobil posunie v smere osi x, tak na začiatku akcelerometer vráti kladné hodnoty podľa veľkosti sily zrýchlenia na ose x. Keď sa mobil zastaví, dochádza ku zníženiu rýchlosti a preto je na ose x namerané záporné zrýchlenie [4].



Obrázek 2.2: Mobilný koordinačný systém

Akcelerometer

Senzor akcelerometer meria zrýchlenie pôsobiace na zariadenie. Na rozdiel od lineárneho akcelerometra meria zrýchlenie nelineárne čo znamená, že namerané hodnoty obsahujú zrýchlenie spôsobené gravitačnou silou pôsobiacou na teleso. Tento senzor určuje zrýchlenie meraním síl, ktoré naň pôsobia. A to podľa nasledujúceho vzťahu [4]:

$$A_d = -\sum \frac{F_s}{m} \tag{2.1}$$

Vzťah medzi gravitačnou silou a meraním zrýchlenia určuje nasledujúca rovnica.

$$A_d = -g - \sum \frac{F_s}{m} \tag{2.2}$$

Kde A_d je pôsobiace zrýchlenie, g je gravitačná sila, F_s je sila zariadenia a m je hmotnosť zariadenia.

Preto pokiaľ by tento senzor bol v pokojnom stave, bez pohybu, stále by nameral zrýchlenie na osách podľa sily gravitácie. Aby bolo zmerané skutočné zrýchlenie, je nutné odstrániť gravitačnú silu z nameraných hodnôt. To sa napríklad docieli cez dolno-priepustný filter.

$$g_{k+1} = \alpha * g_k + (1 - \alpha) * A_d \tag{2.3}$$

Kde α je konštanta, g_{k+1} je vypočítaná gravitačná sila na osi v aktuálnom kroku, g_k je vypočítaná gravitačná sila v predošlom kroku a A_d je hodnota zrýchlenia nameraná akcelerometrom. Táto gravitácia sa následne použije pre výpočet zrýchlenia takto:

$$A_{lin} = A_d - g \tag{2.4}$$

Kde A_{lin} je hodnota zrýchlenia zariadenia bez pôsobenia gravitačnej sily, A_d je hodnota zrýchlenia nameraná akcelerometrom a g je gravitačná sila vypočítana v rovnici 2.3.

Lineárny akcelerometer

Senzor lineárneho akcelerometra poskytuje trojrozmerný vektor, ktorý reprezentuje zrýchlenie na každej osi zariadenia. Toto zrýchlenie neobsahuje pôsobenie gravitačnej sily. Je vhodný pre detekciu gest alebo na implementáciu do inerciálneho navigačného systému. Tento senzor vracia hodnoty podľa nasledujúceho vzťahu [4]:

$$A_{lin} = A_d - A_g \tag{2.5}$$

Kde A_{lin} je namerané lineárne zrýchlenie, A_d je zrýchlenie pôsobiace na teleso a A_g je zrýchlenie pôsobiace na teleso v dôsledku gravitačnej sily.

Kapitola 3

Vizuálna odometria

Táto kapitola popisuje vizuálnu odometriu (d'alej len VO), pretože patrí medzi spôsoby zist'ovania pohybu telesa. Umožňuje zistit' polohu telesa pomocou snímkov z aspoň jednej kamery. Taktiež kapitola obsahuje bližší popis VO s využitím jednej kamery. Pretože mobilné zariadenia zvyknú mat' jednu prednú kameru, ktorá by sa dala použit' pre zistenie polohy pomocou VO.

Úvod Prvý krát bol pojem VO použitý v roku 2004, Nisterom vo svojom orientačnom dokumente. Je to lacná a alternatívna odometrická technika, presnejšia ako GPS, INS a lokalizačné sonarové systémy. Odhad polohy zariadenia je s relatívnou chybou v rozsahu od 0.1% do 2%. V navigácii sa odometria používa, pre odhad zmeny pozície v priebehu času. K tomu sa využívajú údaje zo snímačov pohyb (kamera, senzory a iné). VO je proces počas ktorého dochádza k odhadu polohy určitého agenta (napríklad vozidlo, robot, človek), tým, že sa analyzujú zachytené snímky. Tieto snímky môže snímat' jedna alebo viac kamier [11] [7] [10].

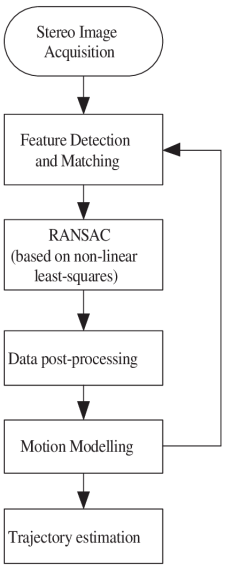
Výhodou VO je to, že na ňu nemá negatívny vplyv nerovnost' terénu, prekĺzavanie kolies, či iné podmienky, ktoré by sa negatívne prejavili pri iných lokalizačných systémoch. V porovnaní s použitím iných senzorov má použitie kamery, pre určenie lokalizácie, výhodu nižších nákladov. To umožňuje jednoduchšiu integráciu určovania polohy v systémoch, ktoré využívajú obrazové videnie, ako napríklad detekcia jazdných pruhov, chodcov alebo prekážok. Fotoaparáty sú pomerne lacné, l'ahké a nemajú vysokú spotrebu energie. Navyše v dnešnej dobe fotoaparát obsahujú skoro všetky mobilné zariadenia. Preto je VO výhodný spôsob, určovania polohy mobilného zariadenia [7].

Hlavné výzvy v systémoch využívajúcich VO, sú oblasti svetelných a zobrazovacích podmienok. Aby VO fungovala efektívne a extrahovala zdanlivý pohyb, je potrebné aby bolo prostredie snímania kamery dostatočne osvetlené a aby mala statická scéna dostatočnú textúru [7] [11].

Vizuálna odometria s využitím jednej kamery Mobilné zariadenia obsahujú väčšinou dve kamery, pričom každá z nich sníma práve opačnú stranu. Preto je nutné zistit' ako funguje VO s využitím práve jednej kamery. V existujúcej literatúre, väčšina navrhnutých metód VO, používa stereo alebo monokulárne kamery. Tým pádom ich je možné klasifikovat' ako stereo systémy VO a monokulárne systémy VO. Monokulárny systém využíva k určeniu polohy telesa práve jednu kameru. Použitie tohto systému znižuje početnost' chýb, ktoré by mohli vzniknút' pri kalibrácii viacerých kamier, počas odhadu pohybu. Tento systém sa

používa najmä v mobilných zariadeniach a notebookoch, kde rozhoduje nízka cena a jednoduché nasadenie [7].

V roku 2004, bola prvý krát prezentovaná rozsiahla VO, ktorá dokázala odhadnúť pozíciu telesa, v reálnom čase s využitím monokulárnej kamery. Používala sledovanie obrysov a náhodný konsenzus vzoriek (RANSAC, anglicky random sample consensus). Nová pozícia fotoaparátu sa počítala cez 3D a 2D odhadu polohy. Vyvinutý algoritmus sa skladal z troch fáz a mohol sa použiť pre monokulárny alebo stereo systém. Prvá fáza algoritmu je detekcia obrysov, druhá sledovanie obrysov a tretia odhad pohybu. Algoritmus začína tým, že na snímkach z kamery, extrahuje rohy a následne zistťuje spoločné prvky medzi jednotlivými snímkami. Následne sa implementuje kritérium zhody, ktorého cieľom je sledovanie spoločných prvkov z jedného snímku na druhý. Nakoniec sa vykoná fáza odhadovania pohybu. V prípade odhadu pohybu monokulárneho systému, sa vypočíta odhad pre každý sledovaný prvok na snímke zvlášť, pomocou algoritmu päťbodovej polohy. Po odhade polohy sledovaných prvkov sa vypočíta ich 3D pozícia s použitím dvoch za sebou idúcich snímkov. Informácie o 3D pozíciach sledovaných prvkov sa použijú pre odhad polohy kamery [7] [11].



Obrázek 3.1: Algoritmus VO s použitím metódy RANSAC [1]

Obrázok 3.1 zobrazuje postup VO stereo systému, od získania snímku z kamery po predikciu polohy zariadenia.

AKAZE KAZE a AKAZE (z anglického názvu accelerated KAZE) sú algoritmy, ktoré sa používajú vo VO. Ako prvé začali používať nelineárnu difúziu vo viacúrovňovej detekcii prvkov na snímkach. Iné algoritmy používajú lineárnu difúziu na tvorbu mierkového priestoru. KAZE a AKAZE vykazujú lepšiu opakovateľnosť a vyšší výkon ako iné algoritmy VO (napríklad ORB, BRISK). Hlavným problémom je náročnosť výpočtu. Pre obrázok s rozlíšením 1024 x 768 je namerané, že detekčná fáza dosahuje v priemere iba 6 fps na procesore Intel Core i7-4790 [12] [13].

[1] Obrázok prebraný z https://www.researchgate.net/figure/General-layout-of-the-visual-odometry-method-based-on-RANSAC_fig5_224329209

Kapitola 4

Testovanie akcelerometrov na platforme Android

Táto kapitola sa zaoberá porovnávaním dvoch typov akcelerometrov, aby sa zistilo, ktorý z nich je lepší pre dosiahnutie čo najlepšieho výsledku pohybu po mape. A vybraný akcelerometer sa viacej skúma.

K tomu, aby sa zistila pozícia mobilného zariadenia, sa potrebuje využit' senzor akcelerometra, pomocou ktorého sa dokáže vypočítat' rýchlost' za čas a tým pádom aj pozícia. Testovali sa dva typy senzorov a to lineárny a nelineárny akcelerometer. Na základe získaných dát sa vybral jeden z nich, ktorý sa využije v implementácii aplikácie. Ten sa zároveň otestoval podrobnejšie na precíznejších testoch, v ktorých sa počíta aj rýchlost' a pozícia zariadenia. Sledujú sa získané hodnoty z akcelerometra a výpočty pozície znázornené na grafoch k tomu, aby sa určil vhodný spôsob ovládania aplikácie.

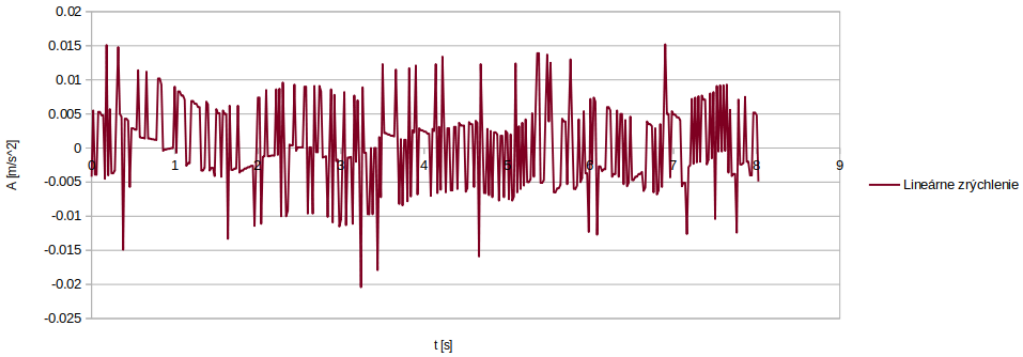
Testovanie prebiehalo v kontrolovaných podmienkach, čo znamená, že sa vie akú veľkú vzdialenost' prešlo mobilné zariadenie. Sledujem aké hodnoty dávajú akcelerometre ($A[m/s^2]$) za čas ($t[s]$ - nelineárny akcelerometer meria približne každých $0.01s$ a lineárny akcelerometer približne každé $0.02s$). Senzory vracajú hodnoty na troch osách x, y a z ako je to na obrázku 2.2. Testovanie prebiehalo na zariadení Huawei P20 Lite (ANE-LX1) s verziou systému 9 (Pie).

4.1 Lineárny a nelineárny akcelerometer

Ako prvé sa porovnáva nelineárny akcelerometer, ktorý vráti zrýchlenie aj so zrýchlením spôsobeným gravitačnou silou a lineárny akcelerometer, ktorý silu gravitácie zanedbáva [4]. Preto je na grafoch nelineárneho akcelerometra vidiet' dve zložky. Nelineárne zrýchlenie a lineárne zrýchlenie. Nelineárne zrýchlenie sú hodnoty zrýchlenia namerané nelineárnym akcelerometrom. Znamená to, že obsahujú zrýchlenie spôsobené gravitačnou silou. Lineárne zrýchlenie sú hodnoty vypočítané pomocou filtru, ktorý je bližšie popísaný tu 2. Počas každého merania sa zaznamenávajú hodnoty získané z lineárneho akcelerometra a zároveň aj z nelineárneho akcelerometra. Teda je vidiet' grafy hodnôt oboch senzorov zvlášt', počas rovnakého testu, za totožných podmienok.

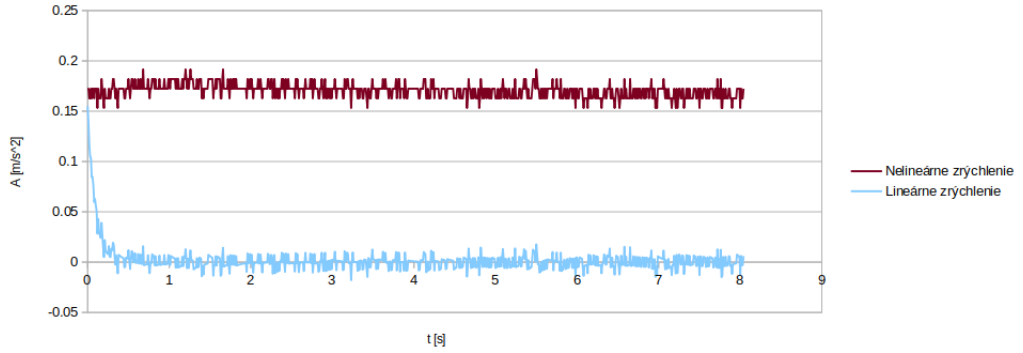
Stabilná poloha

Ako prvý test som zvolil meranie hodnôt oboch senzorov akcelerometrov, počas toho ako je mobil v nehybnej polohe. Pomocou toho môžem určit' základ chovania senzorov. Mobil bol položený na pevnej podložke bez toho, aby naň pôsobila nejaká sila (okrem gravitačnej).



Obrázek 4.1: Lineárny akcelerometer, osa y

Na grafe 4.1 sú namerané hodnoty lineárneho akcelerometra počas doby $8s$. Sú to hodnoty lineárneho zrýchlenia telesa. Maximálna nameraná hodnota dosiahla $0.0152m/s^2$. Minimálna nameraná hodnota $-0.0204m/s^2$. Súčet nameraných hodnôt je $0.1504m/s^2$ a priemerná hodnota je $0,0000376m/s^2$. Z nameraných hodnôt lineárneho akcelerometra vyplýva, že lineárny akcelerometer nepotrebuje počiatočnú inicializačnú dobu, počas ktorej by sa jeho meranie spresňovalo. Pretože namerané hodnoty kmitajú okolo hodnoty $0m/s^2$, čo je vlastne šum. Hodnoty sú v rozpätí od $-0.0204m/s^2$ do $0.0152m/s^2$. Súčet nameraných hodnôt je $0.1504m/s^2$ čo znamená, že pokiaľ by sa počítala rýchlost', tak by bola kladná a postupom času by rástla. Tým pádom by bola aj výsledná pozícia rastúca.



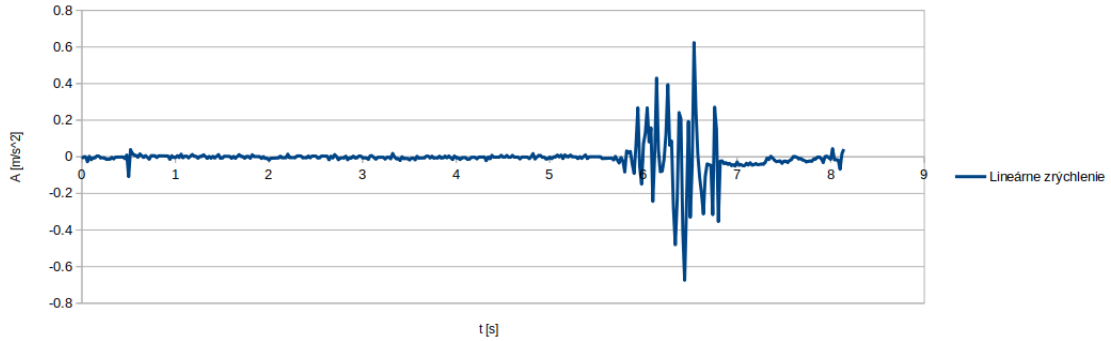
Obrázek 4.2: Nelieárny akcelerometer, osa y

Na grafe 4.2 sú namerané hodnoty nelineárneho akcelerometra, to je nelineárne zrýchlenie. Ďalej je tam vypočítané aj lineárne zrýchlenie, čo je zrýchlenie bez pôsobenia gravitačnej sily. Zameriavam sa na vypočítané lineárne zrýchlenie, pretože to sa potencionálne bude využívat' v implementácii na zistenie rýchlosti a následne pozície. Maximálna nameraná hodnota je $0.155m/s^2$. Minimálna hodnota je $-0.0179m/s^2$. Súčet hodnôt je $1.49m/s^2$ a priemerná hodnota je $0.0039m/s^2$. Súčet hodnôt a priemerná hodnota sú oveľa vyššie čo znamená ešte vyšší odkyv od skutočnej hodnoty ako je to pri lineárnom akcelerometri. Ale rozdiel je v tom, že tento súčet obsahuje aj hodnoty na začiatku snímania senzoru, kedy ešte

nieje stabilizovaná hodnota lineárneho zrýchlenia. Hodnota lineárneho zrýchlenia sa podľa grafu stabilizuje po približne $0.5s$. Súčet hodnôt po tejto dobe je $-0.02255m/s^2$ a priemerná hodnota je $0.00212m/s^2$. Priemerná hodnota po tejto dobe je približne o polovicu nižšia ako pri lineárnom akcelerometri, taktiež absolútna hodnota súčtu je nižšia ako pri lienárnom akcelerometri. To znamená, že pokiaľ by sa vypočítala rýchlosť a pozícia na základe týchto nameraných hodnôt, boli by presnejšie než pri lineárnom akcelerometri. Bola by to pravda len za predpokladu, že sa rýchlosť a pozícia merajú až po $0.5s$ od začiatku snímania týmto senzorom.

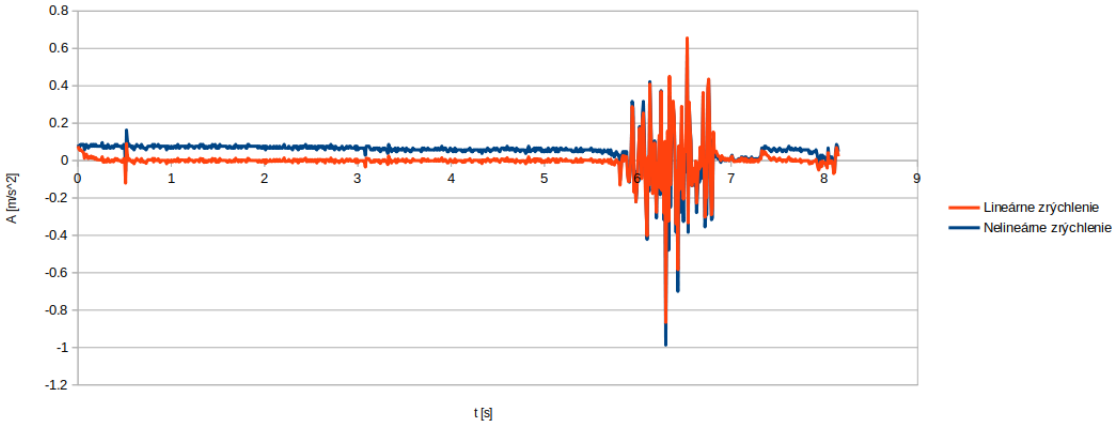
Pomalý pohyb v smere osi x

V ďalšom teste sa zisťuje ako akcelerometre reagujú pri pomalom pohybe zariadenia. Tento test prebiehal tak, že bol mobil položený na pevnej podložke, pričom bol opretý o meradlo s dĺžkou $22cm$. Pozdĺž tohoto meradla, v smere osi x, bol prevedený pomalý pohyb zariadenia. Cele meranie trvalo približne $8s$.



Obrázek 4.3: Lineárny akcelerometer, osa x

Graf 4.3 ukazuje namerané hodnoty lineárneho akcelerometra teda lineárne zrýchlenie. Namerané hodnoty tohto merania dosahovali maximálnej hodnoty $0.6231m/s^2$, minimálnej hodnoty -0.6741, súčtu $-3.093m/s^2$ a priemernej hodnoty $-0.0076m/s^2$. Je vidieť, že graf zrýchlenia nieje ideálny, pretože na začiatku pohybu mobilu mali byť namerané rastúce kladné hodnoty, ktoré začnú klesať k 0, kým je mobil v pohybe. A následne záporné hodnoty pri zastavení mobilu. To znamená, že hodnoty obsahujú pomerne veľa šumu. Súčet nameraného zrýchlenia, by sa v ideálnom stave mal rovnať $0m/s^2$. Tu je súčet $-3.093m/s^2$, čo je vysoký rozdiel.

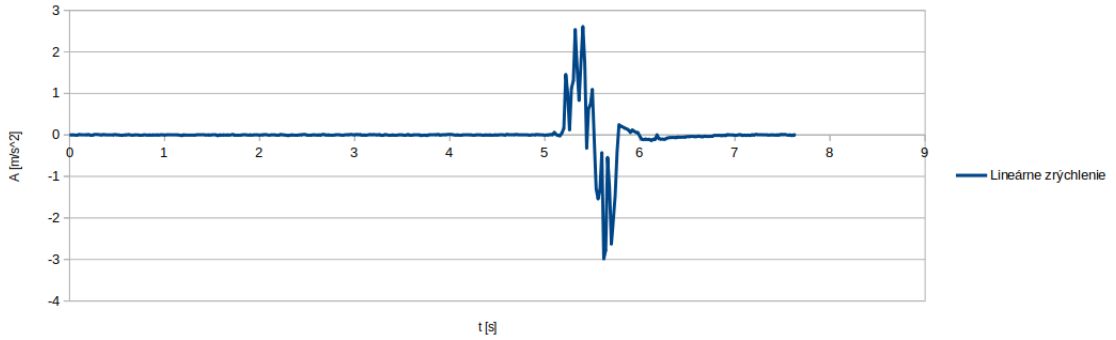


Obrázek 4.4: Nelineárny akcelerometer, osa x

Namerané hodnoty nelineárneho akcelerometra sú zobrazené v grafe 4.4. Zároveň je tam vypočítané aj lineárne zrýchlenie. Teraz ma zaujímajú hodnoty vypočítaného lineárneho zrýchlenia po dobe $0.5s$, pretože to je doba po ktorej sú stabilizované výpočty lineárneho zrýchlenia. Tie dosahujú hodnôt maximálnej $0.655m/s^2$, minimálnej $-0.865m/s^2$ a súčtu $-0.455m/s^2$. Na grafe nieje vôbec vidiet' krivku, akú by malo znázornit' meranie zrýchlenia pohybu do strany. Dáta sa javia ešte viacej zašumené, než pri lineárnom akcelerometri. Výhodou ale je, že súčet hodnôt je oveľa bližšie k nule.

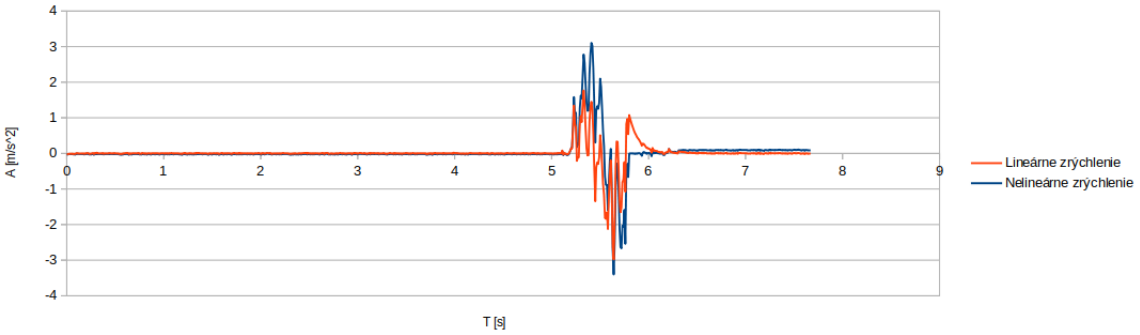
Rýchly pohyb v smere osi x

V tret'om teste získavam namerané dáta senzorov počas rýchleho pohybu. Tretí test prebiehal podobne ako ten druhý. Mobil bol položený na pevnej podložke, pričom bol opretý o meradlo s dĺžkou $22cm$. Pozdĺž tohoto meradla v smere osi x, bol prevedený rýchly pohyb mobilom. Celé meranie trvalo približne $8s$.



Obrázek 4.5: Lineárny akcelerometer, osa x

Graf 4.5 zobrazuje namerané hodnoty lineárnym akcelerometrom. Maximálna hodnota je $2.6m/s^2$, minimálna hodnota je $-2.985m/s^2$ a súčet hodnôt je $-1.558m/s^2$. Tento graf vyzerá ako graf zrýchlenia. Na začiatku pohybu boli namerané kladné hodnoty lineárneho zrýchlenia, keďže sa mobil začal pohybovat' v smere danej osi. Na konci pohybu sa namerali záporné hodnoty keďže zmena rýchlosti bola negatívna.



Obrázek 4.6: Nelineárny akcelerometer, osa x

Na grafe 4.6 sú namerané hodnoty nelineárneho akcelerometra, teda nelineárne zrýchlenie a aj vypočítané hodnoty lineárneho zrýchlenia. Sledujem hodnoty vypočítaného lineárneho zrýchlenia po dobe stabilizačnej dobe $0.5s$. Maximálna hodnota je $1.764m/s^2$, minimálna hodnota je $-2.977m/s^2$, súčet hodnôt je $0.974m/s^2$.

Záver

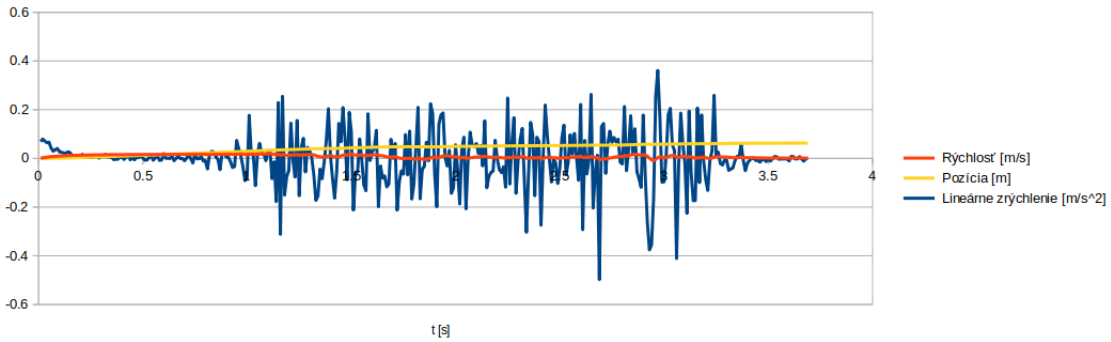
Po zhodnotení dát z nelineárneho a lineárneho akcelerometra som zistil, že oba senzory majú chyby merania. Oba senzory majú pomerne vysoký šum počas stabilnej polohy a aj počas merania pomalého pohybu. Lineárny akcelerometer má výhodu v tom, že vracia hodnoty bez pôsobenia gravitačnej sily. Zatiaľ čo nelineárny akcelerometer potrebuje približne $0.5s$ na stabilný výpočet lineárneho zrýchlenia pomocou dolno priepustného filtru. Po tejto dobe mal ale nelineárny akcelerometer súčet nameraných hodnôt bližší k $0m/s^2$ než lineárny akcelerometer. Ďalším faktorom pri rozhodovaní je aj dostupnosť senzorov, pričom lineárny akcelerometer je menej dostupný (odkaz na tabuľku 2.2). Na základe týchto informácií som sa rozhodol pre získanie lineárneho zrýchlenia zariadenia, použiť nelineárny akcelerometer.

4.2 Nelineárny akcelerometer

Pre využitie v aplikácii som si vybral nelineárny akcelerometer. Sledujem chovanie tohoto senzoru pri rôznych pohyboch zariadenia, aby som dokázal navrhnúť efektívnu manipuláciu s mapou. Najviac ma zaujímajú namerané hodnoty na osách x a y, pretože osa z je kolmá na displej mobilu a s ňou pracovať nebudem. Lineárne zrýchlenie je v jednotkách m/s^2, rýchlosť je v jednotkách m/s a pozícia je v jednotkách m.

Pomalý pohyb v smere osi x

Prvý test prebiehal v kontrolovaných podmienkach. Na pevnej podložke sa mobil posunul pomalým pohybom o $22cm$ v smere osi x. Sledujem vypočítané lineárne zrýchlenie, rýchlosť a či sa vypočítaná pozícia rovná prejdenej. Na grafe 4.7 sú znázornené vypočítané hodnoty

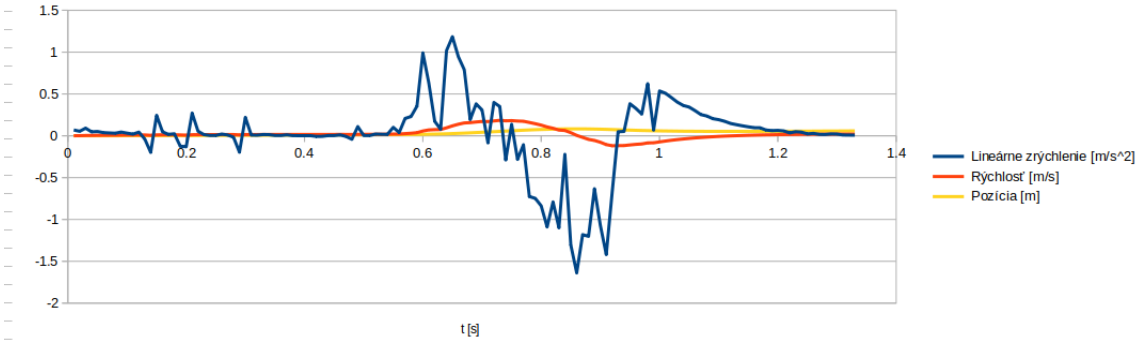


Obrázek 4.7: Pomalý pohyb v smere osi x o $22cm$

lineárneho zrýchlenia, rýchlosti a pozície. Pri pomalom pohybe sú dáta z akcelerometra veľmi zašumené a vypočítaná pozícia je veľmi nepresná. Na grafe vidno, že zmenu pozície v čase robí najmä neustálené počiatočné zrýchlenie zariadenia, ktoré ešte nebolo spresnené pomocou dolno-priepustného filtru. To prinieslo nízku rýchlosť vďaka ktorej sa pozícia časom zvyšovala. S mobilom som prešiel $22cm$ čo znamnená, že vypočítaná pozícia by mala mať hodnotu $0.22m$. Túto hodnotu sa ale nepodarilo dosiahnuť, ani so zvýšenou skreslenou rýchlosťou.

Rýchly pohyb v smere osi x

Druhý test prebiehal v kontrolovaných podmienkach, kedy som s mobilom prešiel $22cm$, v smere osi x rýchlym pohybom. Na grafe 4.8 je vidieť lineárne zrýchlenie, rýchlosť a pozíciu

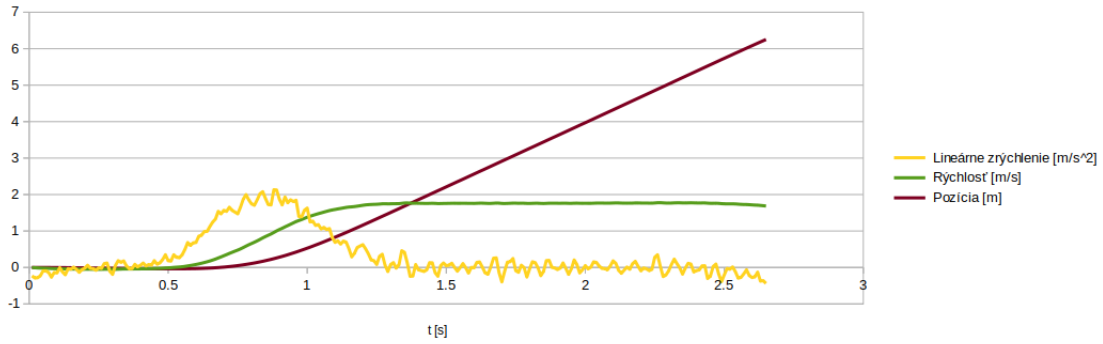


Obrázek 4.8: Rýchly pohyb v smere osi x o $22cm$

počas rýchleho pohybu v smere osi x. Pri rýchlom pohybe sú namerané hodnoty výraznejšie a je vidieť nárast a pokles rýchlosti zariadenia. Na grafe je vidieť že vypočítané hodnoty sú oproti pomalému pohybu vyššie ale pozícia zariadenia sa dostala na nízku hodnotu približne $0.1m$.

Otočenie na osi x

V tretťom teste sledujem ako sa bude chovať senzor a aké hodnoty nameria pokiaľ mobil nebudem do strany hýbať, ale nim iba nakláňať do jednej strany. Test prebiehal tak, že som mobil držal v ruke a otočil ho o 90° do strany. Pri otočení zariadenia na ose x

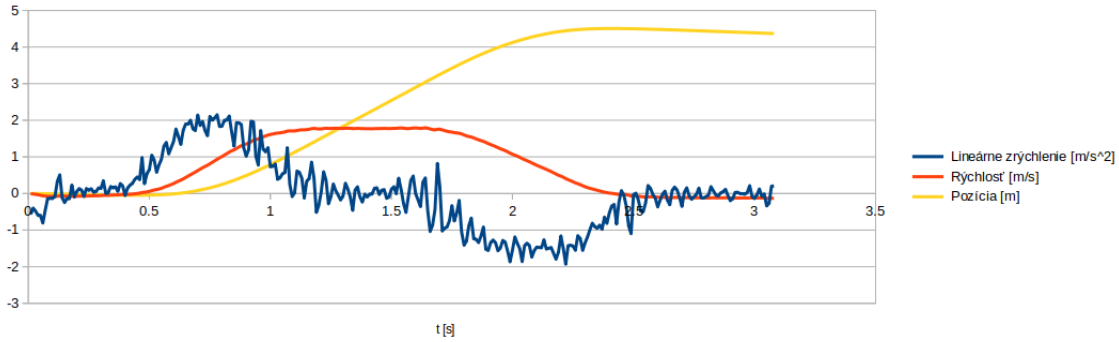


Obrázek 4.9: Otočenie o 90° na osi x v protismeru hodinových ručičiek

v protismeru hodinových ručičiek došlo ku zvýšeniu zrýchlenia k hodnote $2m/s^2$ pričom rýchlosť dosiahla hodnoty $1.9m/s$. Po zastavení pohybu otáčania sa ale nenameralo priamo úmerné záporné zrýchlenie a teda nedošlo k zníženiu rýchlosti. Preto sa konštantne zvyšuje pozícia zariadenia aj napriek tomu, že nieje v pohybu ale iba v náklone.

Otočenie na osi x a naspäť

V štvrtom teste využívam náklon mobilu do jednej strany v ktorej sa aj vraciam. Test prebiehal tak, že som držal mobil v ruke, naklonil ho o 90° do strany a naspäť. Takže sa mobil ocitol v pôvodnej pozícii. Na grafe je vidieť, že sa pri otočení zariadenia v protismere



Obrázek 4.10: Otočenie o 90° na osi x v smere aj protismeru hodinových ručičiek

ručičkových hodiniek zvýšilo zrýchlenie na $2m/s^2$ pričom sa zvýšila rýchlosť na $1.9m/s$, v tom čase rastie pozícia podľa danej rýchlosti. Následne pri otočení zariadenia v smere hodinových ručičiek zrýchlenie kleslo k $-2m/s^2$, a tým sa rýchlosť dostala na $0m/s$, vďaka čomu sa pozícia zastavila na približnej hodnote $4.2m$.

Záver

Testoval som dva typy pohybov. Pohyb mobilom do strán a otočenie mobilu okolo svojej osi. Zistil som, že pri pomalom pohybe do strany v smere osi x je namerané zrýchlenie vysoko zašumené a nepresné pre nameranie pozície. Rýchly pohyb v smere osi x ukazuje výrazne lepšie namerané hodnoty. Pokiaľ by mal užívateľ pohybovať mobilom danou rýchlosťou, ovládanie aplikácie by bolo veľmi nepraktické. Pri otáčaní mobilu okolo svojej osi som zistil, že daný pohyb by sa dal využiť pre ovládanie aplikácie. Pri natočení mobilu sa zaznamenalo kladné zrýchlenie, ktoré kleslo ku $0m/s^2$, po tom, čo som s mobilom zastavil pohyb náklonu. Nenameralo sa ale záporné zrýchlenie, takže bola rovnaká rýchlosť a pozícia konštantne rástla. Keď som s mobilom otestoval aj spätný pohyb náklonu, tak som zistil, že pri spätnom otočení sú namerané záporné hodnoty zrýchlenia. Vtedy sa rýchlosť priblížila k $0m/s$ a následne sa nárast pozície zastavil. Preto je vhodným spôsobom ovládania posunu mapy, pomocou otáčania mobilného zariadenia a nie jeho pohybu do strán.

Kapitola 5

Kalmanov filter

V tejto kapitole popisujem Kalmanov filter, ktorý využívam pre spresnenie výsledkov merania IMU jednotky.

Kalmanov filter je algoritmus, ktorý sa používa pre spresnenie výsledkov merania veličín. Používa radu meraní, nameraných za určitú dobu pozorovania, ktoré zahrňujú šum a iné nepresnosti. Potom ma na výstupe odhady neznámych premenných, ktoré sú zvyčajne presnejšie ako namerané hodnoty veličiny. Tento algoritmus je pomenovaný po Rudolfovi Kálmánovi, ktorý bol jeden z hlavných tvorcov tejto teórie. Tento filter má množstvo aplikácií. Zvyčajne sa používa v navigačných systémoch a riadení strojov, ako sú lietadlá, rakety a lode s dynamickou pozíciou. Ďalšie využitie má v robotike. Využíva sa pre plánovanie pohybu a optimalizáciu trajektórie robotov.

Základný princíp spresnenia výsledkov meraní je v tom, že Kalmanov filter predpovedá stav systému v čase k za predpokladu, že sa tento stav systému vyvinul zo stavu v čase $k-1$. Algoritmus funguje na báze dvoch fáz. Fáza predikcie a fáza aktualizácie. Fáza predikcie produkuje odhad súčasného stavu systému a jeho nepresností. Akonáhle je nameraná hodnota (s určitou nepresnosť ou a šumom) v ďalšom kroku merania, tak sa odhady vo fáze predikcie aktualizujú. K aktualizácií dochádza pomocou váženého priemeru, čo znamená, že každá hodnota ma váhu. Vyššia váha sa dáva odhadom s väčšou pravdepodobnosť ou presnosti [14] [3] [1].

Predikcia Vo fáze predikcie sa počíta predikovaný stav systému. A to presne dve premenné $\hat{x}_k^-$, čo je odhadovaná hodnota a P_k^-, čo je matica chyby kovariancie. V rovnici 5.1 je výpočet predikcie stavu, pomocou použitia modelu dynamického stavu systému, ktorý odhadne jeden krok vpred v čase [14] [2].

$$\hat{x}_k^- = A\ \hat{x}_{k-1} + Bu_{k-1} \tag{5.1}$$

Kde $\hat{x}_{k-1}$ je predošlý odhad stavu, A je matica prechodu stavu, ktorá spája predchádzajúci časový krok s aktuálnym, B je riadiaca vstupná matica aplikovaná na voliteľný riadiaci vstup u_{k-1}.

Ako ďalšie sa vypočíta predikcia matice chyby kovariancie P_k^-, podľa vzťahu 5.2.

$$P_k^- = AP_{k-1}A^T + Q \tag{5.2}$$

Kde P_{k-1} je predchádzajúci odhad chyby kovariancie, A je matica prechodu stavu a Q je kovariancia procesného šumu.

Aktualizácia Vo fázy aktualizácie sa počíta kalmanov prírastok. Kalmanov prírastok sa používa pre aktualizáciu odhadu a aktualizáciu matice chyby kovariancie. Ten sa vypočíta podľa nasledujúcej rovnice 5.3. Informácie som čerpal z [14] [2].

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \tag{5.3}$$

Kde P_k^- je matica chyby kovariancie, H je transformačná matica a R je kovariancia šumu merania. Následne sa prevedie meranie veličiny z_k. Pre aktualizáciu odhadu stavu je nutné vypočítať zbytok merania. Zbytok merania je rozdiel nameranej veličiny z_k a odhadu stavu $\hat{x}_k^-$. Následne sa aktualizuje odhad podľa rovnice 5.4. Kde aktualizácia odhadu sa vypočíta ako odhad zo stavu predikcie plus zbytok merania s váhou kalmanovho prírastku.

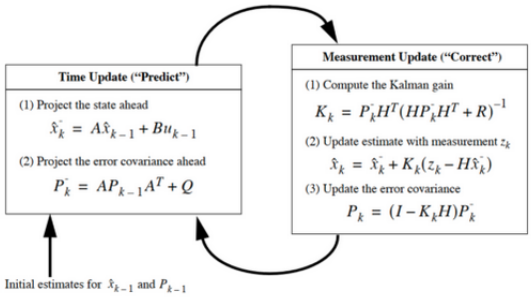
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{5.4}$$

Kde $\hat{x}_k^-$ je odhad, K_k je kalmanov prírastok, H je transformačná matica a z_k je nameraná hodnota. Po získaní aktualizácie odhadu stavu sa kalkuluje aktualizácia matice chyby kovariancie P_k. Podľa vzťahu 5.5.

$$P_k = (I - K_k H) P_k^- \tag{5.5}$$

Kde I je matica identity, K_k je kalmanov prírastok, H je transformačná matica a P_k^- je matica chyby kovariancie.

Zhrnutie Obrázok 5.1 ukazuje obe fázy Kalmanovho filtra aj s rovnicami, ktoré sa v daných krokoch počítajú. Na začiatku je nutné inicializovať filter s odhadom počiatočnej hodnoty stavu a aj jej chyby. Potom sa filtru posielajú namerané hodnoty v každom kroku merania a filter posiela odhad stavu [2].

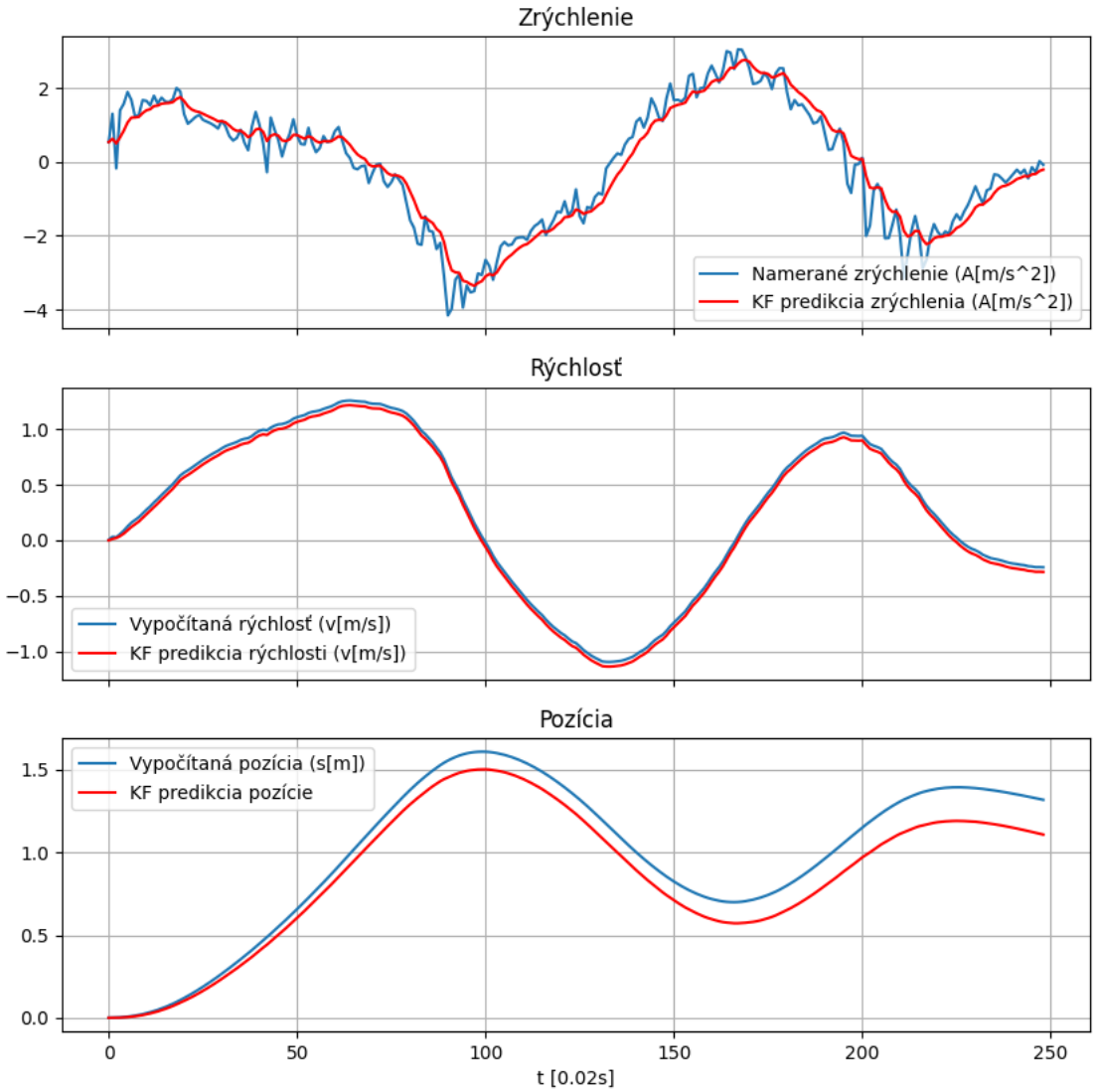


Obrázek 5.1: Fázy Kalmanovho filtra [1]

Testovanie Kalmanovho filtru Po naštudovaní teoretickej časti Kalmanovho filtra, som implementoval testovací kód v Pythone. Inšpiráciu pre implementáciu tohto kódu som čerpal z [8]. Využíva knižnice numpy, matplotlib, csv a pykalman. Numpy sa používa pre prácu s vektormi, maticami a viacrozmernými poľami. Matplotlib je vykresľovacia knižnica pomocou ktorej sa zobrazujú namerané a vypočítané hodnoty v grafoch. Csv je modul, ktorý umožňuje čítať a zapisovať dáta vo formáte csv, čo sú hodnoty oddelené čiarkou (comma-separated values). A knižnica pykalman umožňuje implementovať algoritmus Kalmanovho filtru.

[1] Obrázok prevzatí z https://machinelearningspace.com/object-tracking-python/

Tento kód načítava súbor s už nameranými dátami akcelerometra. Následne inicializuje objekt Kalmanovho filtru a cyklom for prejde celý súbor dát. Pričom v každom kroku cyklu sa pošle namerané zrýchlenie na ose x do objektu Kalmanovho filtru a získa sa odhad zrýchlenia, rýchlosti a polohy zariadenia na tejto ose. Obrázok 5.2 ukazuje 3 grafy. Prvý graf zobrazuje namerané hodnoty zrýchlenia a k tomu predikciu zrýchlenia. Druhý vypočítanú rýchlosť z nameraného zrýchlenia a predikciu rýchlosti Kalmanovým filtrom. A posledný, tretí, vypočítanú polohu z vypočítanej rýchlosti a predikciu polohy Kalmanovým filtrom. Jeden časový krok trval približne $0.02s$.



Obrázek 5.2: Test Kalmanovho filtru, osa x

Simulovaný pohyb je náklon mobilu do oboch strán pozdĺž osi x. Krivka predikcie zrýchlenia filtru je viditeľne menej zašumená a presnejšia. Krivky rýchlosti sa na prvý pohľad príliš nelíšia. Ale graf výslednej pozície ukazuje už pomerne veľký rozdiel, medzi vypočítanou pozíciou a predikovanou pozíciou. Na pozícii je vidieť časom väčší a väčší odklon medzi krivkami. To je spôsobené tým, že pozícia je vypočítaná z pôvodne nameraného zrýchlenia,

ktoré je menej presné. Časom sa tie nepresnosti ešte viac prehlbujú. Zatiaľ čo predikcia pozície Kalmanovým filtrom je počítaná z presnejších hodnôt zrýchlenia a rýchlosti.

Kapitola 6

Návrh mobilnej aplikácie

6.1 Požiadavky na ovládanie pohybu mapy

Výsledkom tejto práce má byť mobilná aplikácia, ktorá bude vedieť zobraziť mapu a jej užívateľ bude schopný vykonať pohyb po mape, iba pomocou pohybu mobilného zariadenia. Ďalej je cieľom vytvoriť interaktívne prvky, aby bola práca s mapou čo najviac intuitívna a efektívna. Pohyb po mape pomocou pohybu mobilného zariadenia sa docieli tým, že sa snímajú dáta zo senzoru akcelerometra, ktorý je dostupný na väčšine mobilných zariadení. Tieto dáta popisujú údaje o zrýchlení daného telesa. Preto je nutné po získaní zrýchlenia, vypočítať rýchlosť telesa a následne novú pozíciu telesa. Novo vypočítanú pozíciu porovnať so starou, zistiť rozdiel a ten animovať na mape v aplikácii.

Následne pre efektívnu prácu s mapou existujú 3 spôsoby ovládania. Z toho dva zahrňujú využitie pohybu mobilného zariadenia. Pre zapnutie a vypnutie týchto spôsobov ovládania mapy, je nutné vytvoriť interaktívne grafické užívateľské rozhranie. Toto rozhranie bude obsahovať tlačidlá, ktoré budú zapínať alebo vypínať jednotlivé funkcie ovládania. Rozhranie bude doplnené o tlačidlá, pomocou ktorých bude možné zvyšovať alebo znižovať rýchlosť pohybu po mape, čo zvýši efektivitu ovládania. Preto aplikácia musí zvládať prijímať požiadavky, pomocou stlačení tlačidiel. A na základe stlačených tlačidiel určiť, o aké ovládanie sa jedná alebo aká rýchlosť pohybu bude použitá. Tlačidlá na rýchlosť pohybu budú dve. Jedno bude znižovať rýchlosť a druhé zvyšovať. Prakticky to znamená, že bude existovať premenná v programe, ktorá bude násobiť výsledný pohyb, ktorý má byť animovaný na mape. Túto premennú budú dané dve tlačidlá meniť a to buď o +1 alebo -1. Minimálne hodnota bude 0. Vtedy bude pohyb po mape nulový, aj keď bude mobilné zariadenie v pohybe. Bude aj maximálna možná veľkosť danej premennej, aby sa predišlo nadmerne vysokej rýchlosti pohybu, ktorá by prakticky nebola účinná.

6.2 Posun mapy

Pri testovaní zrýchlenia z akcelerometra sa testoval rýchly a pomalý pohyb do strán, a taktiež náklon mobilu okolo svojej osi. Zistilo sa, že zrýchlenie ktoré akcelerometer nameral pri pomalom pohybe do strán nebolo vysoké a bolo silno zašumené (4.7). Preto by sa užívateľ musel pohybovať pomocou rýchleho pohybu mobilom, ktorý bol s menším šumom ale je to veľmi nepraktické, pretože vypočítaná rýchlosť dosahovala hodnôt $1 m/s$ (4.8).

Pri otáčaní mobilu okolo svojej osi bolo zrýchlenie menej zašumené. Zároveň pri naklonení o 90° nedošlo pri zastavení otáčania k nameraniu záporného zrýchlenia, čo vedie

k tomu, že rýchlosť ostala konštantná a pozícia taktiež konštantne rástla (4.9). Keď sa ale mobil otočil opačným smerom znova o 90°, tak došlo ku zápornému zrýchleniu počas ktorého sa rýchlosť znížila ku $0m/s$, kedy sa už pozícia skoro nemenila (4.10).

Na základe týchto dát som vytvoril ovládanie mapy tak, že sa užívateľ bude posúvať po mape nakláňaním svojho mobilného telefónu. Existujú tri spôsoby ovládania mapy, z toho dva spôsoby zahrňujú použitie pohybu mobilu a teda využitie senzoru akcelerometra a z nich jeden využíva Kalmanov filter pre spresnenie meraní. Taktiež má užívateľ možnosť zvyšovať a znižovať rýchlosť pohybu mapy. A to kliknutím na dve tlačidlá, ktoré zvyšujú alebo znižujú hodnotu rýchlosti.

Ovládanie bez využitia pohybu zariadenia - dotykom displeja Prvý spôsob ovládania mobilnej aplikácie je predvolený. Užívateľ ovláda mapu dotykom displeja a posunom sa mapa pohybuje. Taktiež dokáže mapu priblížiť alebo vzdialiť v určitom maximálnom rozsahu.

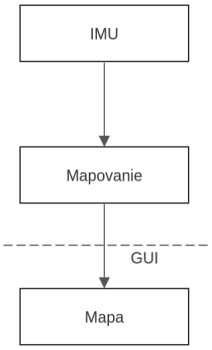
Ovládanie s využitím pohybu zariadenia - náklon Druhý spôsob ovládania pohybu po mape je nazvaný náklon. Tento spôsob sa spustí stlačením tlačidla v aplikácii. Po zapnutí tohoto spôsobu je nutné počkať pol sekundy s mobilom v stabilnej polohe, aby došlo k stabilizácii nameraných hodnôt akcelerometra. Následne, pri naklánaní mobilu do strán sa bude mapa pohybovat'. Je možné zvyšovať alebo znižovať rýchlosť pohybu po mape stlačením ďalších dvoch tlačidiel. Pomocou náklonu sa dá pohybovať po celej mape. Pokiaľ užívateľ nakloní mobil do jednej strany a zastane, mapa sa bude pohybovať daným smerom celú dobu. Až dokým užívateľ neurobí presne opačný pohyb čím sa vynuluje rýchlosť. V ten moment sa pohyb po mape zastaví alebo sa bude pohybovať minimálne, pre nameraný šum a nedokonalosť pohybu mobilom. Tento spôsob pohybu bude využívať Kalmanov filter.

Ovládanie s využitím pohybu zariadenia - rozhľad Tretí spôsob ovládania pohybu po mape je nazvaný rozhľad. Pomenovanie vychádza z toho, že užívateľ sa pohybom mobilu nebude môcť pohnúť po mape viac, než určitý okruh. A stále sa vráti do pôvodnej pozície na mape pokiaľ aj mobil vráti do pôvodnej pozície, od kedy spustil toto ovládanie. Aj pri tomto ovládaní je možné menit' rýchlosť. Tá umožní rozhľad po väčšom okolí. Toto ovládanie nevyužíva Kalmanov filter, pretože danému filtru sa dá v každom kroku namerané zrýchlenie a filter vracia priamo predikciu pozície. Toto ovládanie ale potrebuje počítať vlastnú rýchlosť a následne pozíciu iným spôsobom.

6.3 Softvérový návrh aplikácie

Aplikácia sa skladá z troch hlavných častí, ako popisuje obrázok 6.1: IMU (spracovanie zrýchlenia z akcelerometra), mapovanie (výpočet pozície) a mapa (animácia pohybu po mape a GUI).

IMU Časť IMU sa stará o to, aby aplikácia dokázala spracovávať namerané dáta z IMU jednotky. Jej úlohou je spravovať a riadiť senzor akcelerometra. To zahrňuje vytvorenie manažéra senzorov, prihlásenie senzoru akcelerometra pre príjem nameraných dát, spracovanie nameraných dát z akcelerometra a v prípade, že sa nebude akcelerometer využívať, tak odhlásenie senzoru z príjmu dát. Taktiež bude vedieť zistiť či sa senzor na zariadení, kde sa aplikácia spustí, nachádza a v prípade, že nie, vypíše chybovú hlášku. Pretože prvých $0.5s$ je doba, ktorú potrebuje akcelerometer pre stabilizáciu výpočtu lineárneho zrýchlenia,



Obrázek 6.1: Softverový návrh

tak je nutné aby sa prvú pol sekundu počítalo iba lineárne zrýchlenie. A po pol sekunde sa môže počítať aj rýchlosť, pozícia a zobrazovať pohyb na mape. Bude to mať ten dopad, že rýchlosť nebude skreslená počiatočnou hodnotou zrýchlenia, ktoré nieje stabilizované.

Mapovanie K pohybu mapy v mobilnej aplikácií je potrebné zistiť údaje o polohe zariadenia. Mapovanie prijíma spracované dáta z časti IMU (zrýchlenie, čas). Následne dochádza k predikcii polohy Kalmanovým filtrom, pri ovládaní náklonom a výpočtu rýchlosti a pozície, pri ovládaní rozhľadom. Úzko spolupracuje s grafickým užívateľským rozhraním. Z neho získava informácie o tom ktoré ovládanie mapy je spustené a či sa mení rýchlosť pohybu. Následne posiela informácie o zmene pozícii mape, ktorá animuje pohyb.

V prípade, že je zapnuté ovládanie mapy pomocou pohybu zariadenia, časť IMU spracuje dané hodnoty, ktoré následne pošle na mapovanie. Mapovanie vie ktoré ovládanie je zapnuté a podľa toho vykoná výpočet pozície. Pretože podľa spôsobu ovládania mapy sa líši výpočet pozície. Existujú 3 spôsoby ovládania mapy, z toho dva využívajú pohyb zariadenia: náklonom po celej mape a rozhľadom po okolí (viac popísané v 6.2). Ovládanie spôsobom náklonu po celej mape má výpočet pozície pomocou Kalmanovho filtru. Kalmanov filter potrebuje pri inicializácií odhad počiatočnej polohy zariadenia a odhad chyby meraní. Následne sa volá funkcia filtru, ktorá vráti predikovanú polohu zariadenia. Túto polohu mapovanie porovná s polohou z predošlého kroku a výsledok pošle mape pre animáciu pohybu.

Ovládanie spôsobom rozhľadu po okolí, má výpočet pozície bez použitia Kalmanovho filtru. To najmä z toho dôvodu, že rýchlosť sa počíta inak. Pri tomto ovládaní sa rýchlosť v aktuálnom kroku nesčítava s rýchlosťou z predošlého kroku. To znamená, že keď senzor nameria hodnotu zrýchlenia, tak sa vypočíta rýchlosť iba na základe nameraného zrýchlenia, za čas rovný dĺžke doby od posledného kroku. A to podľa nasledujúcej rovnice 6.1.

$$v_k = a \ * dt \tag{6.1}$$

Kde v_k je rýchlosť zariadenia v aktuálnom kroku výpočtu, a je namerané lineárne zrýchlenie zariadenia a dt je doba počas ktorej bolo zrýchlenie namerané. Po získaní rýchlosti sa vypočíta pozícia podľa vzťahu v rovnici 6.2:

$$p_k = p_{k-1} + v \ * dt \tag{6.2}$$

Kde p_k je pozícia zariadenia v aktuálnom kroku výpočtu, p_{k-1} je pozícia v predošlom kroku, v je rýchlosť a dt je doba počas ktorej sa zariadenie pohybuje danou rýchlosťou.

Ovládania pomocou náklonu a rozhľadu sa teda líšia v tom, že prí ovládaní náklonom sa rýchlosť z predošlého kroku pričíta. To robí Kalmanov filter, odhaduje skutočnú polohu zariadenia. Zatiaľ čo pri ovládaní rozhľadom sa počíta rýchlosť, iba podľa naposledy nameraného zrýchlenia. V konečnom dôsledku to znamená, že pomocou ovládania rozhľadu sa bude môcť užívateľ pohybovať na mape iba v určitom okruhu. S ovládaním náklonom sa bude môcť užívateľ pohybovať po celej mape. Po výpočte novej pozície, sa zistí zmena pozície od minulého kroku a pošle sa mape, ktorá animuje pohyb. Zmena pozície sa zistí podľa nasledujúcej rovnice 6.3. V ktorej dp značí veľkosť zmeny od poslednej pozície, p_k aktuálna pozícia zariadenia a p_{k-1} je pozícia zariadenia z predošlého kroku.

$$dp = p_k - p_{k-1} \tag{6.3}$$

Mapa Ide o celkové užívateľské rozhranie. Stará sa od zobrazenia mapy, po prijímanie udalostí na kliknutí tlačidla. Pri spustení aplikácie inicializuje užívateľské rozhranie. Užívateľské rozhranie pozostáva z mapy a tlačidiel pre výber ovládania a zmenu rýchlosti pohybu po mape. Ďalej rieši delegáciu užívateľských vstupov. Keď je stlačené tlačidlo, tak príjme udalosť a informuje o nej mapovanie. Z mapovania príjme zmenu polohy zariadenia a zobrazí ju ako posun na mape.

Hlavnou funkciou aplikácie je zobraziť mapu. Nevytváral som vlastnú mapu ale použil už existujúci mapový podklad. Vyberal som z viacerých možností na základe rôznych kritérií. Tabuľka 6.1 ukazuje mapové podklady a ich základné vlastnosti.

Mapová služba	**Typ**	**Jazyk**	**Mobilná podpora**	**Dostupnosť**	**Rozšírenia**
Google Maps [1]	SDK	Kotlin	Áno	Platené	Áno
OpenLayers [2]	Knižnica	JavaScript	Áno	Open-source	Áno
Leaflet [3]	Knižnica	Javascript	Áno	Open-source	Áno
OpenStreetMap[4]	Knižnica	JavaScript	Áno	Open-source	Áno
MapBox [5]	SDK	Kotlin	Áno	Platené	Áno

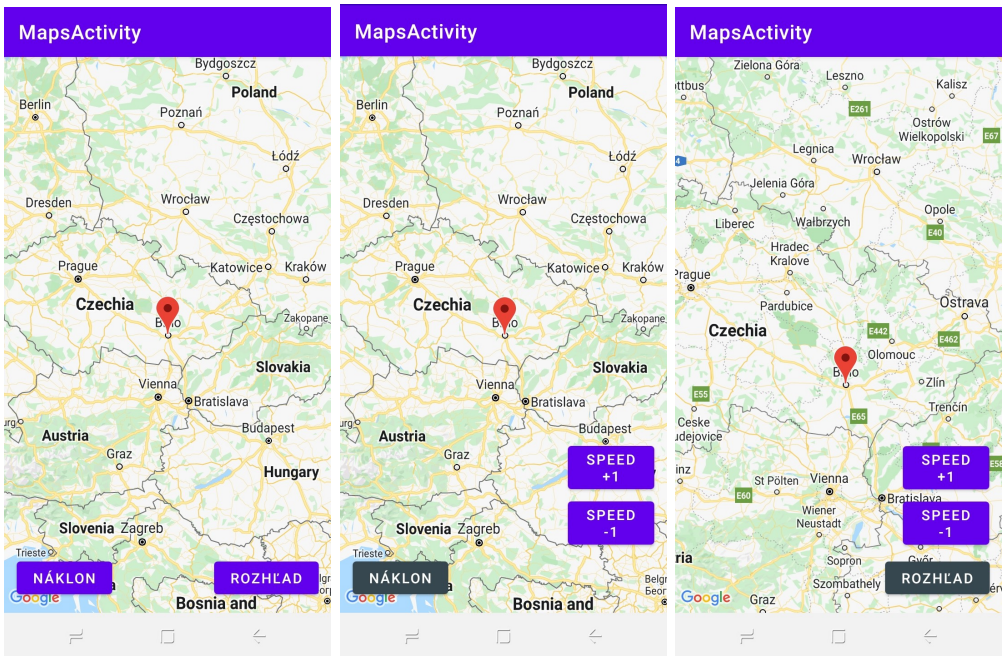
Tabulka 6.1: Mapové služby

Po zhodnotení rôznych mapových služieb som sa rozhodol implementovať Google Mapy. Pretože rovnako ako moja aplikácia, tak aj ona je naprogramovaná v jazyku Kotlin. Má dobré rozhranie a jednoducho sa ovláda. Na internete taktiež existuje veľa návodov, aj od samotnej spoločnosti Google, ako s jej SDK pracovať. Nevýhodou je, že sa za jej použitie platí ale to podľa počtu užití (od 1000 užití a viac). Moje riešenie projektu nebude spustené vysoký počet krát, preto je to zanedbateľný fakt. Taktiež Google Maps umožňujú pohyb po mape pomocou príkazu *CameraUpdateFactory.scrollBy(float xPixel, float yPixel)*. Je to ideálny spôsob pohybu, vzhľadom na to, že môj projekt počíta x a y pozíciu zariadenia. Google mapy je možné jednoducho implementovať v XML súbore Andorid aplikácie cez objekt s názvom *fragment* [5].

[1] Google Maps: https://developers.google.com/maps/documentation/android-sdk/overview
[2] OpenLayers: https://openlayers.org/
[3] Leaflet: https://leafletjs.com/
[4] OpenStreetMap: https://www.openstreetmap.org/
[5] MapBox: https://www.mapbox.com/

6.4 Ovládanie aplikácie a návrh GUI

Po spustení aplikácie sa zobrazí mapa a dve tlačidlá ako je to vidieť na obrázku 6.2. Predvolený spôsob ovládania mapy je dotykom displeja. Ďalšie dva spôsoby ovládania, ktoré využívajú pohyb zariadenia, sa spustia kliknutím na tlačidlá. Po kliknutí na iný spôsob ovládania, sa zobrazia d'alšie dve tlačidlá s ktorými je možné meniť rýchlosť pohybu. Tlačidlá sú umiestnené tak, aby zaberali čo najmenšiu plochu obrazovky. Počas doby ovládania náklonom alebo rozhľadom je dané tlačidlo zafarbené inou farbou aby bolo zrejmé o ktoré ovládanie sa jedná. Zároveň je druhé tlačidlo ovládania skryté, aby nezaberalo zbytočné miesto na obrazovke.



Obrázek 6.2: Obrazovka mobilnej aplikácie

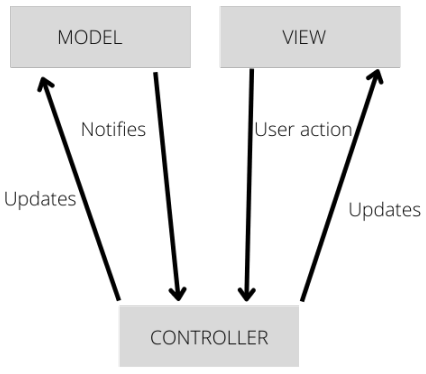
Pre ideálnu interakciu mapy a pohybu mobilu, je nutné držať mobil pol sekundy v statickej pozícii. A to takej, aby bola mobilná obrazovka rovnobežná s tvárou človeka. Počas tej doby sa spresňuje výpočet lineárneho zrýchlenia. Ideálne využitie ovládania rozhľadom je napríklad v prípade, že sa hľadá niečo v okolí pevného bodu. Pri takej situácii je možné prezerať okolie bez toho aby sa v nej užívateľ stratil. Vzhľadom na to, že pokiaľ sa mobil dostane do pozície, pri ktorej bola aplikácia zapnutá, tak aj mapa bude v pôvodnom bode. Počas tohoto spôsobu ovládania je možné zvyšovať rýchlosť pohybu po mape. To umožní užívateľovi prezerať väčšie okolie. Ovládanie náklonom je možné využiť pre prezeranie okolia ako aj pre prezeranie vzdialenejších bodov na mape. Pri náklone mobilu sa rýchlosť sčítava, a tak je v každom kroku výpočtu určitá rýchlosť aj keď je mobil v statickej polohe (ako to ukazuje 4.10).

Kapitola 7

Implementácia mobilnej aplikácie

Cieľom projektu je vytvoriť mobilnú aplikáciu, spustiteľnú na platforme Android. Táto aplikácia musí zobraziť mapu a interaktívne prvky grafického rozhrania, prijímať dáta zo senzoru akcelerometra, počítať potrebné výpočty a zobraziť výsledný pohyb po mape. Implementácia tejto aplikácie je podľa vzoru návrhu aplikácií Model-View-Controller (ďalej len MVC). MVC umožňuje rozdeliť programovú logiku na 3 vzájomne prepojené prvky. To sa robí s cieľom oddeliť informácie zobrazené užívateľovi a interné informácie aplikácie.

- Model (z angličtiny Model) - dynamická dátová štruktúra aplikácie, typicky napríklad databáza
- Pohľad (z angličtiny View) - reprezentácia informácií, zobrazená časť aplikácie užívateľovi
- Kontrolér (z angličtiny Controller) - prijíma vstupy a mení ich na príkazy pre model alebo pohľad



Obrázek 7.1: MVC schéma

Obrázok 7.1 znázorňuje časti MVC a ich vzájomné prepojenie. Pohľad je zobrazený užívateľovi, ktorý s ním interaguje. Pokiaľ užívateľ urobí nejakú zmenu, tak pohľad zašle informáciu o zmene kontroléru. Kontrolér informuje model, ktorý vykoná zmeny na základe užívateľského príkazu. Následne model zašle informácie o zmenách naspäť kontroléru, ktorý ich pošle ďalej do pohľadu.

7.1 Grafické užívateľské rozhranie

Grafické užívateľské rozhranie (z angličtiny Graphic User Interface, d'alej len GUI) je užívateľské rozhranie, ktoré umožňuje užívateľovi ovládat' aplikáciu pomocou interaktivných grafických prvkov. Taktiež sa stará o zobrazenie potrebných informácii užívateľovi v okne aplikácie. GUI patrí, v návrhovom vzore MVC, do časti pohľad, čo znamená že si vymieňa údaje s kontrolérom.

V mobilnej aplikácii sa GUI implementuje formou XML súboru. XML (anglicky eXtensible Markup Language) je značkovací jazyk, ktorý sa používa pre tvorbu dokumentov. Nestará sa o vzhľad, iba o obsah a štruktúru. Umožňuje výmenu údajov medzi užívateľským rozhraním a programom, ktorý beží v pozadí aplikácie. Vzhľad XML súboru sa mení pripojením štýlu. XML súbor v tomto projekte obsahuje dva základné prvky:

- fragment,
- button.

Fragment Fragment je prvok v XML súbore, ktorý umožňuje implementovat' Google mapy do aplikácie. Obsahuje 4 dôležité atribúty:

- *android:id* je identifikátor prvku, pomocou neho s ním dokáže aplikácia pracovat', vymieňat' údaje,
- *android:name* je názov prvku,
- *android:layout_width* je šírka prvku,
- *android:layout_height* je výška prvku.

Jeho kód je nasledujúci:

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/map"
 android:name="com.google.android.gms.maps.SupportMapFragment"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context=".MapsActivity"/>
```

Button Button je interaktivný prvok v XML súbore, ktorý umožňuje prijímať užívateľské vstupy. Rovnako ako fragment aj button obsahuje identifikátor prvku, pomocou s ním aplikácia pracuje a dokáže zistiť, keď dôjde k jeho kliknutiu. Taktiež obsahuje definície šírky, výšky a pozície. V neposlednom rade aj atribút *android:text*, ktorý zobrazí zadaný text ako text tlačidla.

```
<Button
    android:id="@+id/buttonMode2"
    android:layout_width="115dp"
    . . .
    android:text="Rozhl'ad"
    . . .
    />
```

7.2 Akcelerometer

Akcelerometer sa používa pre zisk informácií o zrýchlení telesa. Prístup a získanie dát z akcelerometra umožňuje framework Android senzoru. Framework je súčasťou *android.hardware* balíku a zahŕňa nasledujúce triedy:

- **SensorManager** Používa sa pre vytvorenie instancie senzorovej služby. Pomocou nej sa senzor prihlasuje alebo odhlasuje z príjmania udalostí senzoru. Táto trieda taktiež obsahuje konštanty, ktorými sa určí presnosť merania, určí sa miera obdržavania udalostí alebo kalibrujú senzory.
- **Sensor** Používa sa k vytvoreniu instancie určitého senzoru.
- **SensorEvent** Táto trieda je využitá systémom, ktorý vytvorí objekt typu SensorEvent. Ten obsahuje informácie o udalosti: namerané data, typ senzoru, presnosť meraní a časový odtlačok, kedy k udalosti došlo.
- **SensorEventListener** Používa sa pre vytvorenie dvoch návratových funkcií. Keď sa zmení hodnota senzoru alebo keď sa zmení presnosť.

Pri spustení aplikácie sa ako prvé zavolá inicializačná funkcia *onCreate*. Úlohou tejto funkcie je inicializovať všetky potrebné premenné. V tejto funkcii sa vytvorí instancia triedy *SensorManager* a taktiež instancia triedy *Sensor*, typu *TYPE_ACCELEROMETER*. Následne program čaká na stlačenie tlačidla, ktoré zapne jedno z ovládaní mapy pohybom zariadenia. Stlačením tlačidla sa zavolá funkcia, ktorá zaregistruje senzor akcelerometer na príjem udalostí. Kód tejto funcie je:

```
if(mAccelerometer != null) {
    mSensorManager.registerListener(this, mAccelerometer,
    SensorManager.SENSOR_DELAY_UI)
}
```

Premenná mSensorManager je instancia triedy SensorManager, ktorá zaregistruje a odregistruje senzory z príjmu udalostí. Premenná mAccelerometer je instancia triedy Sensor.

Pri registrácii sa nastavuje aj doba, ako často majú udalosti prichádzať. V tomto prípade SENSOR_DELAY_UI, čo je odporúčaná doba pre aplikácie užívateľského rozhrania. Pokiaľ užívateľ stlačí tlačidlo na vypnutie daného pohybu, tak sa zavolá funkcia, ktorá odhlási akcelerometer z príjmu dát, nasledujúcim kódom.

```
mSensorManager.unregisterListener(this, mAccelerometer)
```

Pokiaľ je senzor zaregistrovaný pre príjem udalostí, tak sú jednotlivé udalosti prijímané funkciou *onSensorChanged(event: SensorEvent?)*. Táto funkcia prijíma objekty typu *SensorEvent*. Dochádza tam aj ku výpočtu pozície a volanie príkazu, ktorý animuje pohyb po mape.

7.3 Výpočet pozície

Pozícia sa počíta pre zistenie polohy zariadenia a pre animáciu zmeny polohy na mape. Existujú dva typy ovládaní, ktoré pracujú s výpočtom pozície, náklon a rozhľad. Náklon zisťuje polohu zariadenia pomocou predikcie pozície Kalmanovým filtrom. Kalmanov filter sa snaží predikovať presnú polohu zariadenia. Rozhľad Kalmanov filter nevyužíva, pretože v tomto pohybe, sa výsledná pozícia nesnaží rovnať presnej pozícii zariadenia. Viacej o návrhu spôsobov ovládania a ich výpočte tu 6.3. Nová pozícia zariadenia sa počíta počas každej príchodzej udalosti senzoru akcelerometra, vo funkcii *onSensorChanged()*.

onSensorChanged(event: SensorEvent?) Funkcia na začiatku skontroluje, či prijatý objekt, *event*, nieje prázdny a či jeho hodnoty niesu nulové. Následne overí typ prijatého objektu. V tomto prípade TYPE_ACCELEROMETER. Je to vhodné najmä v situácii, kedy by bolo v jeden moment zapnutých viacero senzorov. Po skontrolovaní nenulového objektu a typu senzoru prijatého objektu, sa vypočíta zrýchlenie spôsobené gravitačnou silou. Výpočet je v teoretickej rovine popísaný tu 2.3. Prakticky v kóde sa pre výpočet gravitačnej sily používa premenná typu FloatArray [1], ktorá obsahuje vypočítanú gravitačnú silu na osách x a y. Kód výpočtu ne nasledovný:

```
val alpha = 0.9f
gravity[0] = alpha * gravity[0] + (1f - alpha) * event.values[0]
gravity[1] = alpha * gravity[1] + (1f - alpha) * event.values[1]
```

Premenná *event.values[]* obsahuje namerané hodnoty zrýchlenia na osách x, y a z. Po výpočte gravitačnej sily sa vypočíta lineárne zrýchlenie podľa jednoduchého vzorca (2.4). V tomto vzorci sa lineárne zrýchlenie rovná nameranému zrýchleniu od ktorého sa odčíta vypočítané zrýchlenie spôsobené gravitačnou silou. Lineárne zrýchlenie je rovnako uložené v premennej typu FloatArray, ktorá obsahuje vypočítané hodnoty lineárneho zrýchlenia na osách x a y.

```
linearAcceleration[0] = event.values[0] - gravity[0]
linearAcceleration[1] = event.values[1] - gravity[1]
```

[1] FloatArray je pole premenných typu float

Stabilizácia lineárneho zrýchlenia Následne je v kóde podmienka, ktorá má za úlohu odísť z funkcie, pokiaľ ešte neprešlo pol sekundy od registrácie senzoru. To sa robí preto, lebo približne pol sekundy trvá stabilizácia lineárneho zrýchlenia po zapnutí senzoru. Po danej pol sekunde sa už môže počítať výsledná pozícia, ktorá nieje ovplyvnená nestabilným lineárnym zrýchlením. Má kód:

```
if(eventCount < 25){
    eventCount++
    return
}
```

Hovorí, že ak je počet udalostí menší než 25, tak zvýš hodnotu premennej počtu udalostí a odíď z funkcie *onSensorChanged*. Táto podmienka zamedzuje výpočtu pozície a zobrazeniu pohybu po mape. A to z toho dôvodu, že je použitý nelineárny akcelerometer, ktorý meria nelineárne zrýchlenie, to jest zrýchlenie, ktoré nezanedbáva pôsobenie gravitačnej sily. Ako je to popísane tu 4. Udalosti zo senzora chodia priemerne každých 0.02s, po vynásobení číslom 25 to dáva pol sekundy. To je doba potrebná pre stabilizáciu hodnôt lineárneho zrýchlenia. Pokiaľ je premenná počtu udalostí vyššia alebo rovná číslu 25, znamená to, že prešlo približne pol sekundy a môže sa pokračovať vo výpočte. Premenná, ktorá počíta počet udalostí začína od čísla 0. Ďalším krokom funkcie onSensorChanged() je výpočet pozície na základe typu ovládania.

Získanie pozície pri ovládaní náklonom

Pozícia zariadenia sa pri ovládaní náklonom získa pomocou predikcie Kalmanovho filtra. Kalmanov filter je implementovaný v jazyku Python, pre jednoduchšie použitie matíc a výpočetných operácií. Použitie kódu, napísaného v jazyku Python, v Android aplikáciách umožňuje *Chaquopy* plugin [2]. Neobmedzené použitie Chaquopy požaduje licenciu. Pokiaľ licencia nieje zakúpená, tak sa aplikácia po 5 minútach vypne. Licencia v tejto implementácii nieje zakúpená.

Inšpiráciu pre kód Kalmanovho filtru som získal z [2]. Vytvoril som triedu *KalmanFilterClass(object)*, ktorá ma dve metódy. Prvá je *__init__(self)*. Inicializuje potrebné premenné, najmä matice, ktoré využíva Kalmanov filter. A inicializuje objekt Kalmanovho filtru z knižnice pykalman. Druhá metóda je *update(self, acc_val)*. Pri voláni sa jej predáva jeden argument, čo je hodnota nameraného zrýchlenia. V tejto metóde dochádza ku predikcii a aktualizácii. Vracia predikovanú pozíciu zariadenia. Aplikácia vytvára dve instancie tejto triedy, *KalmanFilterClass(object)*. Jednu pre osu x a druhú pre osu y. V každom kroku príjmu udalostí od senzora sa volá funkcia update, pre obe osi.

Získanie pozície pri ovládaní rozhľadom

Zmena času Zmena času sa využíva v rovniciach pre výpočet rýchlosti a pozície, pri ovládaní rozhľadom. Jej výpočet je nasledovný. Objekt *event*, typu *SensorEvent*, obsahuje premennú s názvom *timestamp*. Táto premenná je typu *Long* a vyjadruje čas v nanosekundách, kedy došlo k danej udalosti. Táto premenná monotónne rastie s príchodom nových udalostí pre určitý senzor. Používa sa pomocná premenná *eventLastTimestamp*, ktorá obsahuje hodnotu premennej *timestamp* z predošlej udalosti. Problém nastáva vtedy, pokiaľ je to

[2] https://chaquo.com/chaquopy/

26. udalost'. Pretože premenná *eventLastTimestamp*, ktorá má obsahovať hodnotu posledného času udalosti je nulová. Rozdiel medzi premennými *timestamp* a *eventLastTimestamp* je v takom prípade hodnota premennej *timestamp*. Vzhľadom na to, že priemerná doba príchodu udalostí je 0.02s, je možné porovnať rozdiel premennej *timestamp* s *eventLastTimestamp*. Pokiaľ bude väčší ako 0.02 + 0.08 (tolerancia oneskorenia), nastaví hodnotu premennej zmeny času na 0.02. Premenná *dt* označuje zmenu času a je typu *float*. Čas je pre výpočty potrebné previesť z nanosekúnd na sekundy. Preto rozdiel medzi *timestamp* a *eventLastTimestamp* sa vynásobí číslom 0.000000001, čím sa získa hodnota v sekundách. Kód pre výpočet zmeny času je preto nasledovný.

```
var dt: Float = (event.timestamp - eventLastTimestamp) * NS2S
if (dt > 1f) dt = 0.02f
```

Kde *dt* je doba v sekundách od poslednej udalosti, *event.timestamp* je hodnota času aktuálnej udalosti v nanosekundách, *eventLastTimestamp* je hodnota času predošlej udalosti v nanosekundách a $NS2S$ je konštanta pomocou ktorej sa prevedú nanosekundy na sekundy.

Výpočet rýchlosti a pozície Výpočet rýchlosti a pozície má nasledujúci kód.

```
velocity[0] = linearAcceleration[0] * dt
velocity[1] = linearAcceleration[1] * dt

position[0] += velocity[0] * dt
position[1] += velocity[1] * dt
```

Premenné velocity a position sú typu FloatArray. Obsahujú vypočítanú rýchlosť a pozíciu na osách x a y.

Animácia pohybu po mape

Oba typy ovládaní, náklon aj rozhľad, ukladajú výslednú, vypočítanú pozíciu do premennej *position*. Zmena polohy od poslednej udalosti sa zistí rozdielom aktuálnej pozície od poslednej pozície, ako je to v tomto kóde:

```
dx = position[0] - lastPosition[0]
dy = position[1] - lastPosition[1]
```

Pokiaľ by sa do funkcie na pohyb po mape dal iba výsledok premenných dx a dy, tak by sa mapa skoro nepohla. Pretože sú to veľmi nízke hodnoty. Preto sa používajú premenné *multiplierX* a *multiplierY*, ktorými sa násobí dx a dy, aby bol pohyb na mape výrazný.

```
if (isMode1On) {    // MODE 1 - larger velocity, lower multiplier
    multiplierX = -1200f
    multiplierY = 1600f
} else {            // MODE 2 - smaller velocity, larger multiplier
    multiplierX = -5000f
    multiplierY = 8000f
}
```

Závisí aj na type ovládania. Rozhľad má oveľa nižšiu rýchlosť, pretože sa nesčítava s predošlou rýchlosťou a tak je výsledná pozícia menšia. Preto sa násobí väčším číslom. Hodnota premennej *multiplierX* je záporná, lebo súradnicový systém senzoru a funkcie, ktorá zobrazuje pohyb po mape je rozdielny. Pomocou záporného znamienka sa určí správny smer pohybu v ose x. Nasledujúci kód animuje pohyb po mape.

```
if (dx != 0f || dy != 0f){
    mMap.moveCamera(CameraUpdateFactory.scrollBy(dx * multiplierX
    * speedButtonMultiplier, dy * multiplierY * speedButtonMultiplier))
}
```

Do funkcie *scrollBy()* sa pošlú dva argumenty, veľkosť pohybu na ose x a veľkosť pohybu na ose y. Premenná *speedButtonMultiplier* sa používa pre zvyšovanie a znižovanie rýchlosti užívateľom.

Kapitola 8

Testovanie aplikácie na užívateľoch

8.1 Forma testov

Prvá časť testu je praktická. Aby si v nej tester osvojil aplikáciu, vyskúšal rôzne formy pohybu po mape a otestoval ich na rôznych úlohách. Druhá časť testu je písomná. Kedy po praktickej časti bude testerovi daný dotazník pre vyplnenie, s rôznymi otázkami na jednotlivé typy pohybu a celkovú aplikáciu.

Testovanie prebiehalo osobne, kedy bol každému z 3 testerov predaný papier s popisom úloh a pri ich riešení som sledoval, čo sa im darí alebo nedarí a aké reakcie majú na jednotlivé spôsoby ovládania.

Popis úloh Je 6 úloh pričom každá z nich má podobnú obtiažnosť. Prvých 5 úloh majú testeri dané aký spôsob ovládania k jeho dokončeniu musia využiť. V poslednej úlohe je spôsob ovládania na ich vlastnom uvážení. Úlohy sú pre každého testera v rôznych poradiach až na poslednú. Po dokončení jednotlivých úloh sa užívateľ vracia na pôvodnú pozíciu na mape a to mesto Sydney.

1. Ovládanie pomocou dotyku displeja. Tester ma za úlohu nájsť na mape Teherán následne nájsť na mape Japonsko a nakoniec sa vrátiť do Sydney.

2. Ovládanie pomocou náklonu po celej mape. Tester ma za úlohu nájsť mesto Praha, po jeho nájdení ísť do mesta New York a nakoniec sa vrátiť naspäť do Sydney.

3. Ovládanie pomocou náklonu po celej mape. Tester ma za úlohu prejsť po obvode Nemecka a vrátiť sa naspäť do Sydney.

4. Ovládanie pomocou rozhľadu po okolí. Tester ma za úlohu ísť do mesta Ottawa, následne nájsť na mape Singapúr a nakoniec sa vrátiť do Sydney.

5. Ovládanie pomocou rozhľadu po okolí. Tester ma za úlohu prejsť po obvode Francúzka a vrátiť sa naspäť do Sydney.

6. Ovládanie je voliteľné. Tester ma za úlohu nájsť mesto Wagga Wagga, ktoré sa nachádza v okolí Sydney.

Dotazník Dotazník bol daný testerom po dokončení praktických úloh, v ktorých si vyskúšali všetky typy pohybu po mape v precíznejších úlohách. V dotazníku sú otázky zamerané na konkrétne formy pohybu. Čo sa im na danej forme pohybu páči, aké vidia výhody alebo aké to má nedostatky. Ďalej je tam otázka na ich obľúbené ovládanie, celkový pocit z aplikácie a ohodnotenie jednotlivých foriem pohybu na stupnici. Otázky v dotazníku sú v prílohách tu A.

8.2 Prevedenie testov

Testy boli prevedené osobne, kedy som bol s každým testerom osobitne sám v izbe. Na začiatku som im vysvetlil základné informácie o aplikácií, typy ovládaní a že prvá časť je praktická a druhá formou dotazníku. Následne som testerom podal mobil so spustenou mobilnou aplikáciou a s ním aj zoznám praktických úloh. Testeri mali dostatok času na vypracovanie všetkých úloh, mohli sa pýtať otázky pokiaľ niečomu nerozumeli. Počas doby testovania som sledoval ich reakcie na jednotlivé typy pohybu, čo sa im darí, ako dlho im trvá kým si zvykli na danú formu pohybu, v akých situáciach ju využívali viacej a v akých menej, a iné.

Testeri boli traja. Prvý tester vo veku 20 rokov je študentom pedagogiky, bez vyššej znalosti technológií s obľubou máp. Druhý tester vo veku 28 rokov, pracujúci informatik, kedysi programátor teraz Scrum Master. Tretia testerka vo veku 26 rokov učiteľka na základnej škole bez väčšej znalosti technológií.

Zhrnutie výsledkov Testeri pri výkone úlohy s použitím prvej formy pohybu, dotyku displeja, nemali problém. Všetci testeri sa zhodli na tom, že forma pohybu je precízna a ľahká na ovládanie s tým, že užívateľ má viac pod kontrolou mapu. Testerka číslo 3 podotkla, že ide aj o sílu zvyku, keďže je to bežná forma pohybu vo väčšine mapových aplikáciách. Medzi nevýhody ovládania dotykom displeja sú: pohyb prstom, rýchlosť pohybu po mape je pomalá a nezávisí na rýchlosti dotyku alebo v prípade, že užívateľ má špinavé prsty, tak musí využiť inú časť tela pre dotyk displeja.

Pri ovládaní mapy pomocou druhej formy pohybu, náklonom po celej mape, chvíľu trvalo kým si testeri na danú formu zvykli hlavne keď sa snažili využívať aj tlačidlá pre zmenu rýchlosti pohybu po mape. Najväčšie problémy to robilo testerke číslo 3 ale nakoniec dané úlohy splnila podľa zadania. Medzi výhody ovládania mapy náklonom po celej mape testeri udali, že s využitím tlačidla rýchlosti, je možné sa dostať na vzdialenú časť mapy rýchlo. Ďalej je prechod po mape plynulý. Tester číslo 2 napísal, že ovládanie bolo citlivé a ľahké, pričom sa jednoduchým spôsobom dá zmeniť smer mapy a aj rýchlosť pohybu. Tester číslo 1, ktorý má rád mapy, napísal, že je to zaujímavá forma pohybu po mape a že sa s ňou ešte nikdy nestretol. Tester číslo 2 medzi nevýhody napísal, že užívateľ musí mať pre to cit, nechceným náklonom mobilu sa mapa pohne a najmä pre mobily s malým displejom je náročnejšie sledovať mapu pri ostrom náklone. Tester číslo 1 s tým nemal problémy a testerke číslo 3 prekážal nezvyk, stres.

Tretiu formu pohybu, rozhľad po okolí, si testeri pomerne rýchlo osvojili. Aj pri väčšom náklone sa nestrácali na mape, keďže pri vrátení do pôvodnej polohy sa ocitli znova kde boli predtým. Testerka číslo 3 medzi výhody dala situáciu, kedy musí nájsť niečo v okolí, pričom pomocou tlačidiel na rýchlosť sa okolie „zväčšuje“. Tester číslo 2 napsísal, že výhodou je statický bod na mape, okolo ktorého je nastaviteľný rádius pre rozhľad a keď užívateľ chce tak sa bez problému vráti na pôvodné miesto. Tester číslo 1 napísal, že je to pohodlný

spôsob prezerania okolia. Tester číslo 1 medzi nevýhody napísal, že miera využitia tohoto pohybu je nižšia. Ďalšou nevýhodou je to, že je náročné sa dostať d'alej na mape.

Tester číslo 1 by z daných troch foriem pohybu využíval najradšej náklon po celej mape, pretože je rýchly. Tester číslo 2 by využíval rozhľad po okolí s tým že ku vzdialenejšej destinácií by sa dostal dotykom prstu a v okolí destinácie by využíval rozhľad. Testerka číslo 3 by využívala prst, teda dotyk displeja pretože je na to už zvyknutá. Ale o rozhľade po okolí podotkla, že nieje zlý.

Tester číslo 1 mal celkový pocit z aplikácie milo-pozitívny s tým, že ho prekvapila originálnými funkciami. Tester číslo 2 mal pocit dobrý, napísal: ide ľahko, jednoducho pochopiteľné, rýchly zvyk na ponúknuté formy pohybu. A páčila sa mu možnosť zmeny rýchlosti po mape. Pre testerku číslo 3 to bolo milo prekvapivé.

Na konci mali testeri uviesť od škály 1 (najlepšie) do 5 (najhoršie) ako sa im páčil pohyb náklonom a rozhľadom. Náklon získal v priemere skóre 1.53 periodických. Rozhľad získal skóre 2.3 periodických.

Na záver by som zhodnotil testovanie tak, že každá forma pohybu má svoje pre aj proti. V rôznych situáciách sú vhodné rôzne formy pohybu. Testeri dokázali úlohy zvládnuť bez väčších problémov ale bolo nutné si zvyknúť.

Kapitola 9

Záver

Cieľom tejto bakalárskej práce bolo navrhnúť spôsob ovládania mobilných aplikácii, pracujúcich s mapovými podkladmi tak, aby sa využila znalosť o pohybe mobilného zariadenia v priestore. Preto bolo nutné naštudovať spôsoby, ktorými je možné získať informácie o pohybe zariadenia. Riešenie má byť doplnené o potrebné interaktívne prvky, aby bola práca s mapou čo najviac intuitívna a efektívna. Ďalším cieľom bolo vytvoriť demonštračné riešenie, pomocou vlastnej alebo existujúcej mapovej aplikácie, ktorá bude využívať vhodné existujúce nástroje a knižnice. Následne toto demonštračné riešenie vyhodnotiť na reálnych úlohách a užívateľoch. Posledným cieľom bolo prezentovať kľúčové vlastnosti riešenia, formou plagátu a krátkeho videa.

Výsledkom mojej bakalárskej práce je mobilná aplikácia, ktorá dokáže zobraziť vybraný mapový podklad a umožní jeho užívateľovi pohybovať sa po mape, pomocou pohybu mobilného zariadenia. Po zhodnotení viacerých mapových podkladov, som sa rozhodol použiť Google Mapy, pre implementáciu v mobilnej aplikácii. Navrhol som 3 spôsoby ovládania pohybu po mape. Prvý spôsob je predvolený a nevyužíva pohyb mobilného zariadenia. Ďalšie dva spôsoby využívajú pohyb mobilného zariadenia. Študoval som dva spôsoby, ktorými je možné získať informácie o pohybe zariadenia. Prvý z nich je inerciálna meracia jednotka a druhý je vizuálna odometria. Rozhodol som sa použiť inerciálnu meraciu jednotku, ktorá obsahuje senzor akcelerometru. Tento senzor obsahuje väčšina mobilných zariadení. Pomocou akcelerometru získavam zrýchlenie mobilného zariadenia a z nej počítam pozíciu. Získanou pozíciou dokážem určiť ako sa zariadenie pohybovalo a tým pádom animovať pohyb po mape. Pre jedno z dvoch ovládaní mapy, využívajúcich pohyb zariadenia, som implementoval aj Kalmanov filter. Kalmanovým filtrom som dokázal odstrániť šum a nepresnosti namerané akcelerometrom. Užívateľské rozhranie som doplnil o interaktívne prvky, ktoré zvýšia efektivitu ovládania, zvyšovaním alebo znižovaním rýchlosti pohybu. Kľúčové vlastnosti aplikácie sú prezentované formou plagátu a formou krátkého videa. Plagát je k nájdeniu tu B a krátke video je dostupné tu https://www.youtube.com/watch?v=lixVXURBgnA.

Mobilná aplikácia by sa dala rozšíriť o ďalšie interaktívne prvky, napríklad vyhľadávanie, určovanie pozície alebo zmenu typu mapy (terén, premávka, cyklotrasy a podobne). Ďalším doplnením projektu by mohla byť knižnica. Funkcionalita ovládania mapy by bola súčasťou tejto knižnice, ktorá by umožňovala implementovať ovládanie mapy, pohybom mobilného zariadenia do Android aplikácií.

Literatura

[1] *An Introduction to the Kalman Filter* [[online]]. 2006-07-24. [vid. 2021-08-11]. Dostupné z: https://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf.

[2] *Object Tracking: Simple Implementation of Kalman Filter in Python* [[online]]. 2020-02-15. [vid. 2022-02-03]. Dostupné z: https://machinelearningspace.com/object-tracking-python/.

[3] *Introduction to Kalman Filter* [[online]]. 2022. [vid. 2021-04-01]. Dostupné z: https://www.kalmanfilter.net/background.html.

[4] *Motion sensors* [[online]]. Last updated 2022-05-05. [vid. 2021-11-08]. Dostupné z: https://developer.android.com/guide/topics/sensors/sensors_motion.

[5] *Maps SDK for Android overview* [[online]]. Posledná úprava 2022-05-06. [vid. 2021-11-11]. Dostupné z: https://developers.google.com/maps/documentation/android-sdk/overview.

[6] *Inertial measurement unit* [[online]]. Posledná úprava 21. april 2022. [vid. 2021-11-06]. Dostupné z: https://en.wikipedia.org/wiki/Inertial_measurement_unit.

[7] Agel, M. O. A., Marhaban, M. H., Saripan, M. I. a Ismail, N. B. Review of visual odometry: types, approaches, challenges, and applications. *Sprinder*. 2016, zv. 5, č. 1897.

[8] Anton. [[online]]. 2018-01-03. [vid. 2022-04-01]. Dostupné z: https://stackoverflow.com/questions/47210512/using-pykalman-on-raw-acceleration-data-to-calculate-position.

[9] B., S. a O., K. Handbook of Robotics. In:. Berlin: Springer, 2008, kap. Sensing and Perception. ISBN 978-3-540-23957-4.

[10] D., N., O., N. a J., B. Visual odometry. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* 2004, sv. 1, s. I–I. DOI: 10.1109/CVPR.2004.1315094.

[11] Davide, S. a Friedrich, F. Visual Odometry [Tutorial]. *IEEE Robotics Automation Magazine*. 2011, zv. 18, č. 4, s. 80–92. DOI: 10.1109/MRA.2011.943233.

[12] F., D. Stereo tracking and three-point/one-point algorithms: a robust approach in visual odometry. In: *IEEE International Conference on anonymous image processing, 2006. IEEE, Piscataway.* 2006, s. 2777–2780.

[13] Lester, K., Khaled, M. a Diana, G. Accelerated Embedded AKAZE Feature Detection Algorithm on FPGA. In:. Jún 2017, s. 1–6. DOI: 10.1145/3120895.3120898.

[14] Milan, S., Vaclav, H. a Roger, B. Image Processing, Analysis, and Machine Vision. In:. 2. vyd. Pacific Grove, CA 93950, USA: Brooks/Cole Publishing Company, 1999, kap. Motion analysis, s. 709–710. ISBN 0-534-95393-X.

[15] N., A., Ghazilla a Khairi. *Reviews on Various Inertial Measurement Unit (IMU) Sensor Applications* [[online]]. December 2013. [vid. 2021-11-08]. Dostupné z: http://www.ijsps.com/uploadfile/2013/1128/20131128022014877.pdf.

Prílohy

Príloha A

Dotazník

1. Čo sa vám páči alebo aké vidíte výhody na ovládaní mapy pomocou dotyku displeja?
2. Čo sa vám nepáči alebo aké vidíte nevýhody na ovládaní mapy pomocou dotyku displeja?
3. Čo sa vám páči alebo aké vidíte výhody na ovládaní mapy náklonom mobilu?
4. Čo sa vám nepáči alebo aké vidíte nevýhody na ovládaní mapy náklonom mobilu?
5. Čo sa vám páči alebo aké vidíte výhody na ovládaní mapy rozhľadom po okolí?
6. Čo sa vám nepáči alebo aké vidíte nevýhody na ovládaní mapy rozhľadom po okolí?
7. Ktorý spôsob ovládania vám najviac vyhovuje a prečo?
8. Aký máte celkový pocit z aplikácie?
9. Ohodnoďte ovládanie mapy pomocou náklonu mobilu od 1 do 5, pričom 1 je najlepšie a 5 najhoršie.
10. Ohodnoďte ovládanie mapy pomocou rozhľadu od 1 do 5, pričom 1 je najlepšie a 5 najhoršie.

Príloha B

Plagát

Obrázek B.1: Plagát

Príloha C

Obsah pamäť ového média

Pamäť ové médium obsahuje nasledujúce zložky:

1. Aplikácia - obsahuje všetky súbory aplikácie
2. Aplikácia APK - obsahuje APK súbor aplikácie a README
3. Dátová sada - obsahuje namerané údaje z testovania akcelerometrov
4. Kalmanov Filter - obsahuje zdrojový kód Kalmanovho filtru použitého pre jeho testovanie spolu s dvoma dátovými súbormi
5. Plagát - obsahuje plagát
6. Technická správa - obsahuje všetky potrebné súbory technickej správy spolu s vygenerovaným pdf
7. Video - obsahuje video prezentácie aplikácie