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 **YEHOR POHREBNIAK**
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: **Pohrebniak Yehor**
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

Práce je zaměřena na vytvoření mobilní aplikace, která představuje mapu a na základě pohybu mobilního zařízení v prostoru provádí změnu polohy mapy. Teoretická část je zaměřena na seznámení s dostupnými technologiemi a návrh systému pomocí metod zpracování dat z inerciálních měřících jednotek mobilního zařízení a videa z kamery chytrého telefonu. Praktická část je zaměřena na implementaci mobilní aplikace, která získá data ze senzorů pohybu a provede posun mapy.

Abstract

This thesis focuses on the creation of a mobile application that presents a map and based on the movement of a mobile device in space, represents a sliding map camera. The theoretical part is focused on acquaintance with available technologies and design of a system of data processing methods from inertial measuring units of mobile devices and video from the smartphone camera. The practical part is focused on the implementation of mobile application that obtain data from motion sensors and move the map.

Klíčová slova

IMU, vizuální odometrie, homografie, mapa, mobilní aplikace, Kalmanův filtr, akcelerometr, zpracování dat, Android, Google Maps.

Keywords

IMU, visual odometry, homography, map, mobile application, Kalman filter, accelerometer, data processing, Android, Google Maps.

Citace

POHREBNIAK, Yehor. *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.

.....
Yehor Pohrebniak
17. května 2022

Poděkování

Chtěl bych poděkovat vedoucímu mé práce, panu Ing. Vítězslavovi Beranovi Ph.D, za jeho cenné rady, doporučení, trpělivost, čas, cenné poznámky a připomínky, a také za možnost častých osobních konzultací při vedení mé bakalářské práce. Dále chci poděkovat své rodině a přátelům za podporu.

Obsah

Kapitola 1

Úvod

V dnešní době jsou uživatelská rozhraní velmi důležitou částí lidského života ve světě technologií. Cílem uživatelského rozhraní je umožnit efektivní ovládání stroje různými způsoby z lidské strany, zatímco stroj poskytuje zpětné informace pro další obsluhu. Příkladem této koncepce může být komunikace mezi člověkem a mobilním zařízením, což dnes ve svém životě používá skoro každý.

Uživatelská rozhraní hrají důležitou roli v chytrých mobilních aplikacích, které mají velký význam pro uživatele. Současné chytré telefony dovolují různé metody komunikace s nimi pomocí senzorových displejů, zvuku, mikrofonu, kamery, řady senzorů. Tyto jednotky otevírají velký počet způsobů komunikace uživatele s aplikacemi. Jednou z takových aplikací je obyčejná mapa, kterou používá většina lidí.

Cílem této práce je navrhnout způsob používání mobilní aplikace mapy. Hlavní kritérium je umožnit prohlížení mapy pomocí pohybu mobilního zařízení obyčejnému uživateli chytrého telefonu. Aplikace musí mít základní funkcionalitu podobnou aplikací mapy. Způsob ovládání mapy je postaven na čtení dat ze senzorů, které jsou dostupné V dnešních chytrých telefonech. Práce se zabývá načtením a zpracováním těchto dat, způsobem reprezentace pohybu mapy a technologiemi pro implementaci výsledné aplikace.

Čtenář se seznámí s dostupnými technologiemi pro řešení problému, s problematikou zadání a s procesem vývoje výsledné aplikace. Tato práce se skládá z několika částí. V teoretické části se čtenář seznámí se základy nutnými pro lepší pochopení problematiky vývoje aplikací s využitím pohybu zařízení a získání a zpracování dat pohybu. V kapitole o návrhu je představen předpokládaný návrh aplikace s použitím technologií z teoretické části. Návrh řeší obecný princip budoucí implementace, možnost výběru implementačního rozhraní, řadu možných způsobu implementací mapy, architekturu, způsob zpracování a získávání dat pro optimalizací výsledného řešení. Dále je popsána podrobná implementace výsledné aplikace na mobilním zařízení, kde se čtenář dozví o tom, jaké technologie byly použity, jak je implementované zpracování přicházejících dat a výsledná architektura aplikace. Navíc je v kapitole, která je věnovaná implementaci, popsána vybraná varianta implementace mapy a programátorského rozhraní pro implementaci. Poslední kapitola je věnovaná testování výsledné aplikace na uživatelích, experimentům, pomocí kterých byl zvolen obecný způsob použití, implementaci programu a testování jednotlivých částí.

Kapitola 2

Teoretický úvod

Tato kapitola uvádí čtenáře do problematiky práce metodou seznámení s potřebnými informacemi pro další pochopení návrhu a implementace. Konkrétně popisuje metody získávání dat pro odhad pohybu zařízení a jejich zpracování. Souřadnice zařízení lze určit pomocí vizuální odometrie a inerciální měřící jednotky. Důležité je pochopit, že získaná data jsou potřebná pro ovládání mapy pomocí pohybu mobilního zařízení, proto se zde popisuje technologie pro řešení daného problému. Zpracování začátečních dát provádí Kalmanův filtr pro odstranění šumu ve výsledné aplikací. Kalmanův filtr se používá v řadě programů, které se zabývají dynamickým výpočtem polohy pomocí přicházejících dát. Dále jsou zde popsána dostupná rozhraní pro návrh a implementaci mobilních aplikaci.

2.1 Vizualní odometrie

Vizuální odometrie je proces odhadu polohy daného prostředku, který je založen na zpracování sekvence obrazových dat zachycených pomocí jedné či více kamer zařízení. Při pohybu kamery se detekuje změna na základě jednotlivých snímků. Díky tomu lze určit aktuální polohu zařízení.

Termín vizuální odometrie byl vytvořen v roce 2004 Nisterem v jeho orientačním dokumentu [2]. Termín byl zvolen pro svou podobnost s odometrií kol, která inkrementálně odhaduje pohyb vozidla integrací počtu otáček jeho kol v průběhu času. Podobně vizuální odometrie funguje tak, že postupně odhaduje polohu vozidla prostřednictvím zkoumání změn, které pohyb vyvolává na snímcích jeho palubních kamer. Aby vizuální odometrie fungovala efektivně, mělo by být v prostředí dostatečné osvětlení a statická scéna s dostatečnou texturou, aby bylo možné extrahovat zdánlivý pohyb. Kromě toho by po sobě jdoucí snímky měly být zachyceny tak, aby bylo zajištěno dostatečné překrývání scén.

Výhodou vizuální odometrie vzhledem k odometrii kol je, že vizuální odometrie není ovlivněna prokluzem kol v nerovném terénu nebo jinými nepříznivými podmínkami. Bylo prokázáno, že ve srovnání s odometrií kol poskytuje vizuální odometrie přesnější odhady trajektorie s relativní chybou polohy v rozmezí od 0,1 do 2,0 %. Díky této schopnosti je vizuální odometrie zajímavým doplňkem k odometrii kol a navíc k dalším navigačním systémům, jako je globální polohovací systém (GPS) nebo inerciální měřicí jednotky (IMU). V dnešní době odometrii lze použít i místo jednotky GPS – má výhodu, že nemusí přijímat signál.

Všechny základní metody vizuální odometrie můžeme rozdělit do dvou skupin: monokulární a stereo metody. Hlavní podmínkou pro přidělení metod do jedné ze skupin je počet použitých kamer.

Monokulární metoda

Odlišnost monokulární metody oproti stereo je v tom, že pro realizaci se používá jedná kamera. Především, monokulární metoda má svou nevýhodu oproti stereo: je závislá na škálovacím faktoru a při zpracování dat jsou nutné získat i data z ostatních senzorů (např. IMU). U monoculární odometrií prostorová projekce je přepočítávaná z 2D dat [2]. Možný postup pro monokulární metodu [14] pomocí odhadu pohybu z 3D na 2D je popsán v následujících krocích:

1. Extrahujte body v prvním snímku F_1 v časovém kroku I, přiřaďte jim deskriptory.
2. Extrahujte body v dalším snímku F_{I+1} a přiřaďte jim deskriptory.
3. Spojte body mezi dvěma po sobě jdoucími snímky. Odhadněte transformaci (s předdefinovaným měřítkem) mezi prvními dvěma snímky pomocí „5-point“ algoritmu [9] a pomocí této transformace spojité odpovídající body.
4. Extrahujte body v následujícím snímku F_{I+2} a spojte je s dříve extrahovanými body z předchozího snímku.
5. Použijte RANSAC k upřesnění shod a odhadněte transformaci, která dává minimální součet čtvercových rozdílů mezi pozorovanými body v aktuálním snímku F_{I+2} a znovu promítnutými 3D body, které byly rekonstruovány ze dvou předběžných. Tento proces se nazývá algoritmus perspektivních N bodů (PnP) [8].
6. Spojte odpovídající dvojice bodu mezi F_{I+1} a F_{I+2} do 3D bodů pomocí odhadované transformace.
7. Nastavte $I = I + 1$, opakujte od kroku 4 pro každou iteraci.

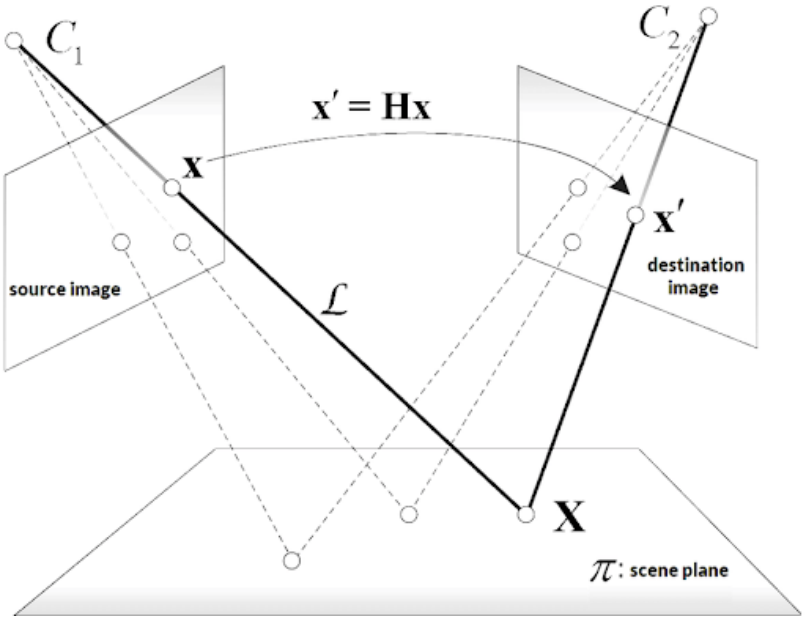
Kalibrace kamery

Cílem kalibrace kamery je přesné změření vnitřních a vnějších parametrů kamerového systému. Ve vícekamerovém systému (např. stereo a trinokulární) vnější parametry popisují vzájemnou polohu a orientaci mezi každým párem kamer. Nejoblíbenější metoda využívá plošný šachovnicový vzor. Pozice polí na herním plánu je známá. Pro přesný výpočet kalibračních parametrů musí uživatel pořídit několik snímků desky zobrazené v různých polohách a orientacích, přičemž zajistí, aby bylo zorné pole kamery co nejvíce vyplněno. Vnitřní a vnější parametry jsou pak nalezeny pomocí metody minimalizace nejmenších čtverců. Vstupní data jsou 2D polohy rohů čtverců desky a jejich odpovídající pixelové souřadnice v každém obrázku.

2.2 Homografie

V oblasti počítačového vidění lze transformovat jakékoliv dva obrazy stejného rovinného povrchu v prostoru, které jsou spojené pomocí homografií. To má mnoho praktických aplikací, jako je oprava obrazu, registrace obrazu nebo pohyb kamery -– rotace a posun -– mezi

dvěma snímky. Jakmile byl proveden posun kamery z odhadnuté matice, lze tyto informace použít pro navigaci nebo pro vložení modelů 3D objektů do obrázku nebo videa tak, aby byly vykresleny se správnou perspektivou a vypadaly, jako by byly součástí původní scény 2.1.



Obrázek 2.1: Grafické zobrazení principu homografií[1].

Princip homografie

Předpokládáme bod (**x, y**), který je zobrazen na ploše počítače. Bod (**x, y**) promítneme do neznámého bodu na projekčním plátně. Tento bod se promítne do neznámého bodu (**X, Y**) do obrazu kamery. Cílem homografie je získat zpětnou transformaci, kde při znalosti pouze bodu (**X, Y**), získáme jejím uplatněním souřadnicí bodu (**x, y**).

V této situaci existují dvě roviny: plocha počítače a obraz kamery. Mezi nimi existuje projektivní transformace, ze které lze vypočítat homografní matici H. Pokud máme tuto matici, pomocí násobení lze vypočítat souřadnice (**x, y**) z bodu (**X, Y**).

Nechť vektory p_a a p_b obsahuji souřadnice bodů (**X, Y**) a (**x, y**) a matice H reprezentuje homografní matici.

$$p_a = \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}, p_b = \begin{bmatrix} \omega x \\ \omega y \\ \omega \end{bmatrix}, \mathbf{H} = \begin{bmatrix} h_1 1 & h_1 2 & h_1 3 \\ h_2 1 & h_2 2 & h_2 3 \\ h_3 1 & h_3 2 & h_3 3 \end{bmatrix} \tag{2.1}$$

Platí, že

$$p_b = \mathbf{H} p_a \tag{2.2}$$

Pro získání matice H je potřeba vědět alespoň čtyři body z obou rovin [11]. Důležitá podmínka je, že žádný ze třech bodů nesmí ležet v rovině na jedné přímce. Obecně lze použít čtyři až N bodů. Pomoci těchto bodů sestavíme matici:

[1]Převzato z: https://www.researchgate.net/figure/Illustration-of-homography-induced-a-by-planar-scene-and-b-by-purely-rotational_fig1_335039112

$$H = \begin{bmatrix} -X_1 & -Y_1 & -1 & 0 & 0 & 0 & x_1X_1 & x_1Y_1 & x_1 \\ 0 & 0 & 0 & -X_1 & -Y_1 & -1 & y_1X_1 & y_1Y_1 & y_1 \\ -X_2 & -Y_2 & -1 & 0 & 0 & 0 & x_2X_2 & x_2Y_2 & x_2 \\ 0 & 0 & 0 & -X_2 & -Y_2 & -1 & y_2X_2 & y_2Y_2 & y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -X_n & -Y_n & -1 & 0 & 0 & 0 & x_nX_n & x_nY_n & x_n \\ 0 & 0 & 0 & -X_n & -Y_n & -1 & y_nX_n & y_nY_n & y_n \end{bmatrix} \tag{2.3}$$

Pro úpravu matice lze využít princip Gaussovy eliminační metody. Soustavu lineárních algebraických rovnic lze vyjádřit rozšířenou maticí soustavy. Nejprve převádíme tuto matici do tvaru, kdy se pod hlavní diagonálou nachází pouze nuly, pomocí řádkových úprav. Upravená matice pak odpovídá soustavě rovnic, která je ekvivalentní s původní soustavou. Matici tedy upravíme na trojúhelníkový tvar a odpovídající body pak použijeme pro sestrojení samotné matice homografie 3x3.

2.3 Random Sample Consensus — RANSAC

Metoda **Random Sample Consensus** zkráceně **RANSAC** se používá pro mapování a porovnávání jednotlivých lokálních oblastí nalezených v obraze mezi sebou ve snaze vyhodnotit jejich správnost na základě jejich podobnosti. RANSAC je založena na zpracování a zjednodušení dat při použití jiných metod, jako například pomocí Bag of words. Metoda RANSAC vede k zavedení chyb v nepřesnosti. Metoda určuje nejlepší řešení pro nalezení detekované oblasti a jejich porovnání mezi sebou. Mezi jednotlivými obrazovými příznaky je metoda schopná najít jejich podobnosti. Princip metody je sestaven na řazení do skupiny iterativních algoritmů, kdy samotný výpočet algoritmu probíhá v jednotlivých iteracích. RANSAC detekuje klíčové oblasti jednoho obrázku, které porovná s detekovanými oblastmi jiného obrázku. Klíčové oblasti porovná náhodně, což umožňuje najít nejoptimálnější řešení modelu pro porovnání. Na základě oblastí vyhodnocuje, zda si jsou nebo nejsou podobné. Podobností pak prohlašuje za dobré nebo špatně shody a počítá jejich maximální a minimální vzdálenosti. Obrázky s větší podobností dostávají přednost před ostatními obrázky. Dále to umožňuje nalezení nejlepšího modelu pro porovnání a jeho volbu, aby výsledek porovnání pomocí homografie byl co nejlepší [3].

RANSAC dokáže zmenšit nepřesnost systému způsobenou zavedením chyb jinými metodami. Rychlost metody závisí na počtu detekovaných oblastí v jednotlivých fotografiích, počtu kandidátů a náročnosti výpočtu porovnání. Nevýhodou této metody je nutnost zadání počtu kandidátů neboli zadání počtu obrázků, mezi kterými se bude porovnávat.

2.4 Pohybové senzory

Inerciální měřicí jednotka (IMU) je elektronické zařízení, které měří a hlásí specifickou sílu těla, úhlovou rychlost a někdy i orientaci těla pomocí kombinace akcelerometrů, gyroskopů a někdy magnetometrů. IMU se obvykle používají k řízení v letadlech (polohový a směrový referenční systém), včetně bezpilotních vzdušných prostředků (UAV), mezi mnoha jinými, a kosmických lodí, včetně satelitů a přistávacích modulů. Nedávný vývoj umožňuje výrobu zařízení GPS s podporou IMU.

Inerciální měřicí jednotka funguje tak, že detekuje lineární zrychlení pomocí jednoho nebo více akcelerometrů a rychlost otáčení pomocí jednoho nebo více gyroskopů. Některé

zahrnují také magnetometr, který se běžně používá jako referenční směr. Typické konfigurace obsahují jeden akcelerometr, gyroskop a magnetometr na osu pro každou ze tří hlavních os (víz. 2.2): sklon, natočení a vybočení [12].

V této práci je důležité se zaměřit na senzorové pohyby současných mobilních zařízení, které mají velmi podobný princip získávání dat pro návrh a implementaci aplikace. V další podsekci jsou popsány různé typy pohybových senzorů dostupných na platformě **Android**.

Pohybové senzory pro mobilní telefony

Platforma Android poskytuje několik senzorů, které umožňují sledovat pohyb zařízení. Možné architektury senzorů se liší podle typu senzoru, což se dále probírá podrobněji.

Senzory gravitace, lineárního zrychlení, vektoru rotace, významného pohybu, počítadla kroků a detektoru kroků jsou buď hardwarové, nebo softwarové. Senzory akcelerometru a gyroskopu jsou vždy hardwarové. Většina zařízení se systémem Android má akcelerometr a mnoho z nich nyní obsahuje gyroskop. Dostupnost senzorů založených na softwaru je variabilnější, protože při odvozování dat často spoléhají na jeden nebo více hardwarových senzorů. V závislosti na zařízení mohou tyto softwarové senzory odvozovat svá data buď z akcelerometru a magnetometru, nebo z gyroskopu.

Pohybové senzory jsou užitečné pro sledování pohybu zařízení, jako je náklon, chvění, rotace nebo houpání. Pohyb je obvykle odrazem přímého vstupu uživatele (například uživatel řídí auto ve hře nebo uživatel ovládá míč ve hře), ale může být také odrazem fyzického prostředí, ve kterém zařízení sedí (například pohyb s vámi při řízení auta). V prvním případě sledujete pohyb vzhledem k referenční soustavě zařízení nebo referenční soustavě vaší aplikace; ve druhém případě sledujete pohyb vzhledem ke světovému referenčnímu rámci.

Všechny senzory pohybu vracejí vícerozměrná pole hodnot senzorů. Například během jedné události senzoru akcelerometr vrací data o síle zrychlení pro tři souřadnicové osy a gyroskop vrací data o rychlosti otáčení pro tři souřadnicové osy.

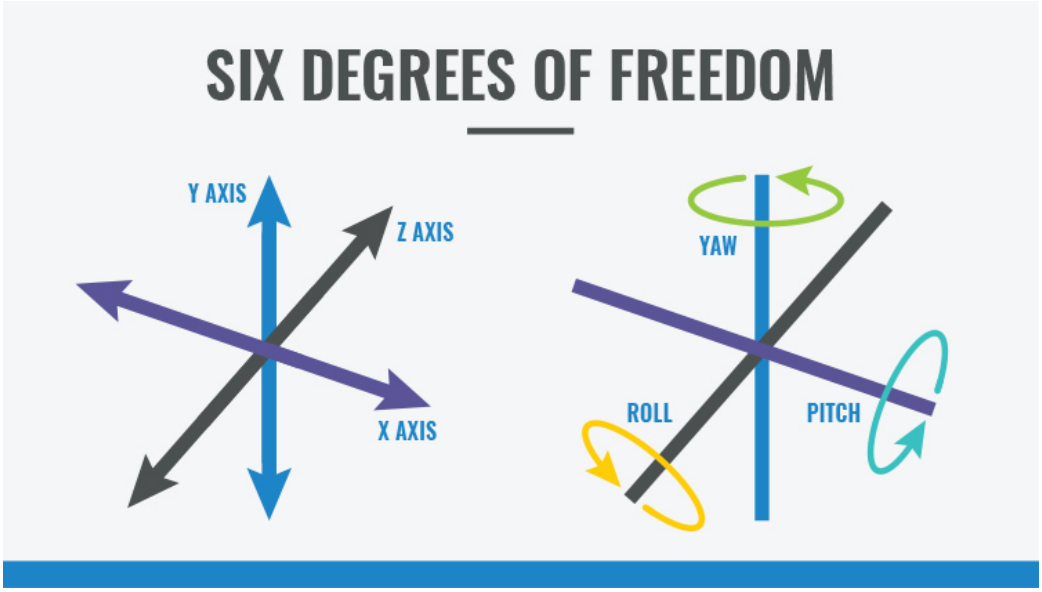
Senzor zrychlení měří zrychlení působící na zařízení, včetně gravitační síly. Akcelerometry používají standardní senzorový souřadnicový systém. V praxi to znamená, že když zařízení leží na stole naplocho ve své přirozené orientaci, platí následující podmínky:

- Pokud zařízení zatlačíte na levou stranu (takže se přesune doprava), hodnota zrychlení souřadnice x je kladná.
- Pokud zařízení zatlačíte na spodní stranu (takže se od vás oddálí), hodnota zrychlení souřadnice y je kladná.
- Pokud zatlačíte zařízení směrem k obloze se zrychlením s hodnotou A m/s^2, hodnota zrychlení z se rovná $A + 9,81$, což odpovídá zrychlení zařízení (A m/s^2) mínus gravitační síla Země (- 9,81 m/s^2).
- Stacionární zařízení bude mít hodnotu zrychlení +9,81, což odpovídá zrychlení zařízení (0 m/s^2 minus gravitační síla Země, která je -9,81 m/s^2).

Obecně platí, že akcelerometr je dobrý senzor, který lze použít, pokud běží monitorování pohybu zařízení. Téměř každý telefon a tablet se systémem Android má akcelerometr a spotřebuje asi 10krát méně energie než ostatní senzory pohybu. Jedinou nevýhodou je, že je potřeba implementovat filtr pro odstranění gravitační síly a snížení šumu.

Gyroskop měří rychlost rotace v rad/s kolem třech os zařízení. Souřadnicový systém senzoru je stejný jako u senzoru zrychlení. Rotace je kladná proti směru hodinových ručiček.

To znamená, že pozorovatel, který se dívá z nějakého kladného místa na souřadnicích x, y nebo z na zařízení umístěné na počátku, by hlásil pozitivní rotaci, pokud by se mu zařízení zdálo rotující proti směru hodinových ručiček. Toto je standardní matematická definice kladné rotace a není stejná jako definice náklonu, kterou používá senzor orientace. Obvykle je výstup gyroskopu integrován v průběhu času pro výpočet rotace popisující změnu úhlů v průběhu časového kroku.



Obrázek 2.2: Reprezentace souřadnicového systému senzorů pohybu na mobilním zařízení[3].

2.5 Kalmanův filtr

Kalmanův filtr je jedním z nejdůležitějších a nejběžnějších algoritmů pro odhady. Kalmanův filtr vytváří odhady skrytých proměnných na základě nepřesných a nejistých měření. Kalmanův filtr také poskytuje předpověď budoucího stavu systému na základě minulých odhadů [6].

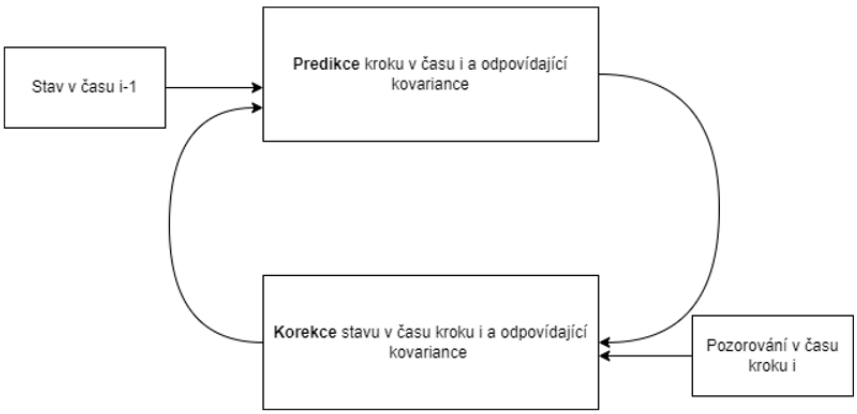
Filtr je pojmenován po Rudolfu E. Kálmánovi (19. 5. 1930 – 2. 7. 2016). V roce 1960 Kálmán publikoval svůj slavný článek popisující rekurzivní řešení problému lineárního filtrování diskrétních dat [13].

Kalmanova filtrace má četné technologické aplikace. Běžnou aplikací je navádění, navigace a řízení vozidel, zejména letadel, kosmických lodí a lodí s dynamickým modelem [10]. Kromě toho je Kalmanovo filtrování konceptem, který se často používá v analýze časových řad používaných pro témata, jako je zpracování signálů a ekonometrie. Kalmanovo filtrování je jedním z hlavních témat plánování a řízení robotického pohybu a lze jej použít pro optimalizaci trajektorie. Vzhledem k časové prodlevě mezi vydáním motorických příkazů a přijetím senzorické zpětné vazby poskytuje použití Kalmanových filtrů realistický model pro vytváření odhadů aktuálního stavu motorického systému a vydávání aktualizovaných příkazů.

Algoritmus pracuje dvoufázově 2.3. Pro fázi predikce Kalmanův filtr vytváří odhady proměnných aktuálního stavu spolu s jejich nejistotami. Jakmile je pozorován výsledek dal-

[3]Převzato z https://www.ceva-dsp.com/ourblog/what-is-an-imu-sensor/

šího měření (nezbytně narušený nějakou chybou, včetně náhodného šumu), tyto odhady se aktualizují pomocí váženého průměru, přičemž větší váha se přikládá odhadům s větší jistotou. Algoritmus je rekurzivní. Může pracovat v reálném čase pouze za použití současných vstupních měření a stavu vypočítaného dříve a jeho matice nejistoty; nejsou vyžadovány žádné další minulé informace [1].



Obrázek 2.3: Princip KF.

Iterace Kalmanového filtru můžeme popsat pomoci následujících rovnic:

Popis chování systému:

$$\begin{aligned} \dot{x} &= \mathbf{A}x + \mathbf{B}u \\ z &= \mathbf{H}x \end{aligned} \tag{2.4}$$

Predikce:

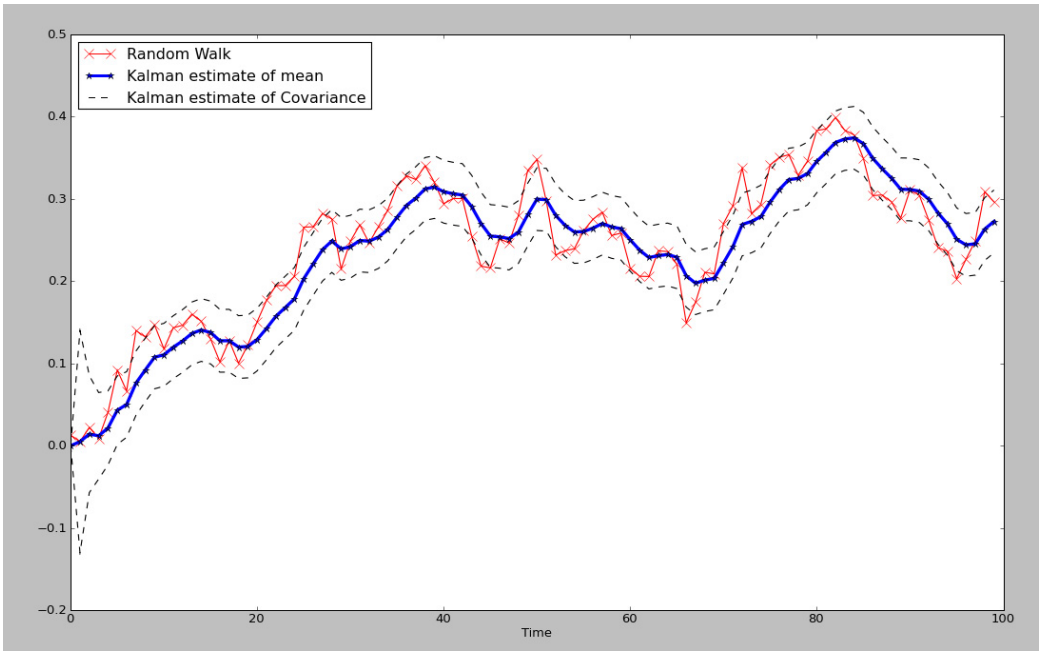
$$\begin{aligned} x_{k+1} &= \mathbf{A}x + \mathbf{B}u \\ \mathbf{P}_{k+1} &= \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q} \end{aligned} \tag{2.5}$$

Korekce:

$$\begin{aligned} \mathbf{K} &= \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1} \\ x_{k+1} &= x_k + \mathbf{K}(z_k - \mathbf{H}x_k) \\ \mathbf{P}_{k+1} &= (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k \end{aligned} \tag{2.6}$$

Kde,
$\mathbf{A}$ - přechodový model, $\mathbf{B}u$ - model řídicího vstupu, H - pozorovací matice, x_k - odhad pozic dynamického modelu, $\dot{x}$ - pozice dynamického modelu, $\mathbf{P}_k$ - kovarianční matice chyby, $\mathbf{Q}$ - kovarianční matice šumu procesu, $\mathbf{K}$ - Kalmanův zisk, $\mathbf{R}$ - kovarianční matice šumu měření.

Grafické znázornění výsledku Kalmanova filtru je zobrazeno na obrázku 2.4. V grafu je reprezentace pozic, která je zašuměná měřením. Výsledkem filtrace je odhad pozice s korekcí a současně je dobře vidět, že šum je mnohem menší než původní.



Obrázek 2.4: Příklad výsledku Kalmanova filtru[4].

2.6 Prostředky pro vývoj mobilních aplikací

Mobilní aplikace je softwarový program vytvořený speciálně pro chytré telefony. V dnešní době jsou chytré telefony integrální částí životů milionů uživatelů. Mezí vývojáři jsou populární speciální prostředky pro implementací mobilních aplikací. Takové prostředky umožňují zjednodušit vývoj a testování přímo na chytrých telefonech. Celá práce se věnuje vývoji aplikace na telefon, proto se v této sekci čtenář seznámí s daným problémem. V této sekci se popisují některé prostředky pro vývoj mobilních aplikací na různých platformách. Cílem je vyhodnotit možná prostředí a vybrat na základě toho nejvhodnější pro danou problematiku.

Android Studio

„Android Studio“ je oficiální integrované vývojové prostředí (IDE) pro operační systém Android od společnosti Google, postavené na softwaru IntelliJ IDEA společnosti JetBrains a navržené speciálně pro vývoj systému `Android`. Je k dispozici ke stažení na operačních systémech `Windows`, `MacOS` a `Linux` nebo jako služba založená na předplatném v roce 2020. Jedná se o náhradu za Eclipse Android Development Tools (E-ADT) jako primární IDE pro vývoj nativních aplikací pro Android.

7. května 2019 `Kotlin` nahradil `Javu` jako preferovaný jazyk Google pro vývoj aplikací pro Android. `Java` je stále podporována, stejně jako `C++`.

React Native

Open-source framework „React Native“ je úzce spojen s javascriptovou knihovnou `React`, ze které přejal většinu jejích vlastností. Obě technologie pocházejí z dílny Facebooku a jsou typické svou svižností a nezávislými znovupoužitelnými komponentami, jež do značné míry zrychlují a usnadňují vývoj webových i mobilních aplikací.

[4]Převzato z http://intelligenttradingtech.blogspot.com/2010/05/kalman-filter-for-financial-time-series.html

React Native je postaven na specifické syntaxi `JSX`, tedy na `JavaScriptu`, který je překládán do `Javy`, nebo `Objective-C`. Můžeme tak vytvářet plnohodnotné nativní aplikace pro iOS i Android. Prostřednictvím nativních komponent, které nahradily původní webové bloky, pak pracuje s API stejně jako nativní kód a má tak na rozdíl od hybridních aplikací větší přístup k hardwaru zařízení.

Xamarin

„Xamarin" patří mezi nejstarší technologie určené pro multiplatformní vývoj mobilních aplikací. Již nějakou dobu spadá do portfolia amerického softwarového giganta Microsoft, což mu dává obrovskou výhodu v podobě snazší integrace například s cloudovým řešením Azure.

K vývoji v `Xamarinu` využíváme programovací jazyk `C#`, který je stejně jako JSX u React Native překládán do nativních jazyků pro Android a iOS. Na rozdíl od Native však Xamarin můžeme použít nejen pro multiplatformní mobilní aplikace, ale i pro nativní vývoj, a to jak pro Android a iOS, tak pro desktopové aplikace pro Windows.

Závěr

V této sekci byly představeny vybrané prostředky pro vývoj mobilních aplikaci. Důležitou částí této práce je výběr operačního systému, na kterém bude prováděno testování a implementace programu. Operační systém hraje velkou roli při návrhu výsledné aplikace, zejména dostupnost openCV v implementačním prostředí. Pří návrhu je důležité prozkoumat prostředky pro vývoj mobilních aplikací a zjistit, které z nich umožňují komunikovat s hardwarovými jednotkami mobilu, konkrétně se senzory pohybu. Cílem práce je také implementovat mapu a soustředit se na realizací ovládání mapy pomocí pohybu mobilního zařízení, proto prostředek musí podporovat vhodné openCV mapy.

Android Studio je velmi používaný a pokročilý prostředek, který odpovídá požadavkům pro výběr implementačního rozhraní. Android Studio nabízí řadu dostupných openCV a možnost výběru programovacího jazyka. Při vývoji softwaru nevznikají problémy s testováním a debugováním aplikace. Z jiné strany React Native umožňuje implementovat na více platformách, než pouze Android. React Native má své výhody a nevýhody oproti Android Studio. Xamarin není tak populární mezi vývojáři, jako Android Studio a React Native, proto je na webu a v knihách menší množství dostupných informací, než u jiných prostředí. Pro danou problematiku jsou nejvhodnější Android Studio nebo React Native.

2.7 Uživatelská zkušenost

V této sekci se probírá problém spojený s realizací výsledné aplikace. Program předpokládá nový pohled pro použití mapy na mobilním zařízení, proto je zaměření na použitelnost aplikace z pohledu uživatele neoddělitelnou částí práce. Zde se čtenář seznámí s pojmem „Uživatelská zkušenost" a se vztahem tohoto terminu s problematikou daného vývoje.

UX (User Experience) – je termín známý jako uživatelská zkušenost. Tento pojem se zabývá aspekty lidské zkušenosti se systémem. Pojem zahrnuje grafiku, design, rozhraní, fyzickou interakci a manuál. Dále se tento termín vykládá jako snaha o vytvoření soudržného, prediktivního a především žádoucího designu.

UX se často spojuje s termínem „usability". Tento termín znamená použitelnost – pro UX je důležité, aby věc, kterou vyprodukujeme, byla použitelná. UX se často spojuje s termínem, který s nim souvisí – UI – „User Interface", ale tyto termíny mají různý význam a roli [7].

Zkušenost uživatele se systémem může ovlivnit mnoho faktorů. Za účelem řešení rozmanitosti byly faktory ovlivňující uživatelskou zkušenost rozděleny do tří hlavních kategorií: stav uživatele a předchozí zkušenosti, vlastnosti systému a kontext použití (situace) [5]. Porozumění reprezentativním uživatelům, pracovní prostředí, interakce a emoční reakce pomáhají při navrhování systému.

Přehlednost a dostupnost informací a akcí je v aplikacích klíčová. Uživatel očekává, že systém nebude náročný na použití. GUI (Graphical User Interface – grafické uživatelské rozhraní, obsahuje interaktivní grafické ovládací prvky) a design hrají v této problematice velkou roli. V této práci je důležité, aby GUI s designem nebyly náročné pro uživatele.

Uživatel potřebuje mít možnost pohybovat mapou nejen pomocí pohybu mobilního zařízení v prostoru, ale i klasickou metodou, protože když bude mít potřebu posunout mapu hodně daleko, nemusí potahovat mobilem na stovky metrů nebo otáčet telefon do polohy, ve které není vidět displej. Je nutné se zamyslet nad tím, jak realizovat vhodné a srozumitelné GUI pro tento problém.

Kapitola 3

Návrh aplikace

V teto částí je popsán způsob návrhu aplikace pro prohlížení mapy metodou pohybu zařízení. Pro úspěšné dosažení cílů je potřeba udělat několik úloh, které pomůžou implementovat výslednou aplikaci:

1. Navrhnout způsob získávání dat z inerciálních měřících jednotek.
2. Navrhnout způsob získávání dat pohybu z kamery.
3. Navrhnout zpracování získaných dat pohybu z kamery a inerciálních měřících jednotek.
4. Navrhnout odstranění šumu pomocí Kalmanova filtru.
5. Navrhnout způsob implementace mapy.

3.1 Analýza úlohy

Cílem je navrhnout a vytvořit aplikaci pro prohlížení mapy „neklasickým způsobem“ metodou pohybu zařízení. Typický uživatel této aplikace je člověk s chytrým telefonem, který využívá mobilní aplikace pro zjednodušení svého života. Zpravidla uživatel potřebuje podobné aplikace v případech hledání různých objektů (adres, obchodů atd.) nebo zjištění svojí geopozice na mapě.

V dnešní době většina lidí používá ve svých chytrých telefonech chytré aplikace s metodou komunikace přes senzorový displej. Klasickou mapou (např. Google Map/Mapy Seznam) jsme zvyklí pohybovat pomocí gest na displeji (např. přiblížení se většinou provádí rozvedením prstů nebo posun vlevo se provádí pohybem prstu zleva doprava). V této práci je hlavním cílem implementovat alternativní způsob komunikace mezi uživatelem a programem pro experiment, zda taková metoda bude užitečná nebo ne.

Potřeby uživatele

Uživatel může mít telefon s sebou kdekoliv a kdykoliv. Dnes je možně připojit zařízení k internetu skoro všude, proto aplikace může vyžadovat internet. Hlavním požadavkem uživatele je intuitivní rozhraní a stabilní program, který bude mít ladné prohlížení mapy. Je důležité mít dostatečný posun mapy vzhledem k pohybu chytrého telefonu, což znamená, že vývoj aplikace musí být dobře zaměřen na odstranění šumu a nastavení správných koeficientů v relaci dat pohybových senzorů a přemisťování mapy. Přitom je potřeba uživatelům vysvětlit, jak používat tuto novou metodu prohlížení mapy pro dosažení cílů.

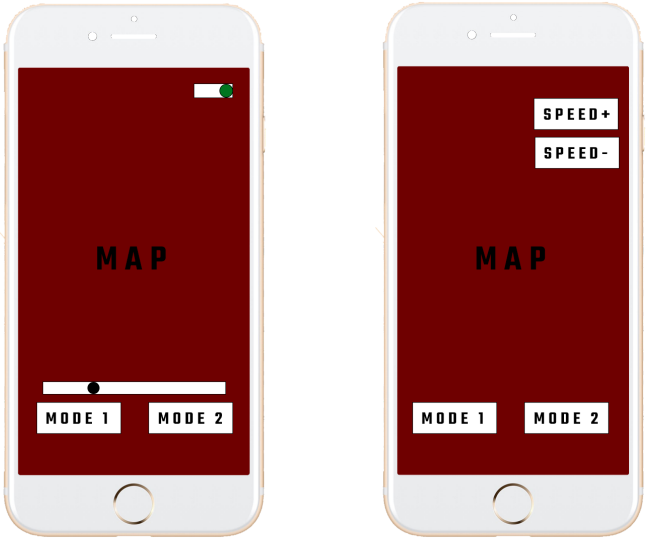
Analýza problému

Návrh výsledné aplikace představuje klasickou mapu, na kterou jsme zvykli v běžném životě. Může obsahovat základní funkcionalitu jako GPS, názvy ulic, objektů a pole pro vyhledávání objektů na mapě. Hlavním cílem je implementovat vhodné řešení pro sběr dat pomocí knihoven a zpracovávat tato data algoritmem pro odstranění šumu. Zpracovávaná data je potřeba převést na správnou změnu kamery mapy v aplikaci. Konečné řešení musí mít souřadnice změny $[d_x, d_y, d_z]$, kde d_x je pohyb mapy vpravo, nebo vlevo, d_y – nahoru nebo dolů, d_z – zvětšení nebo zmenšení mapy. Mapa se musí měnit hned po pohybu, proto je třeba se zaměřit na použité algoritmy, aby aplikace byla dostatečně optimalizovaná a rychlá. Hlavní problémy jsou:

- Správné a rychlé získání dat
- Algoritmus pro zpracování dat z IMU a kamery
- Koeficienty, které budou použity u algoritmů
- Dobrá architektura aplikace
- Užitečné uživatelské rozhraní
- Aktualizace mapy

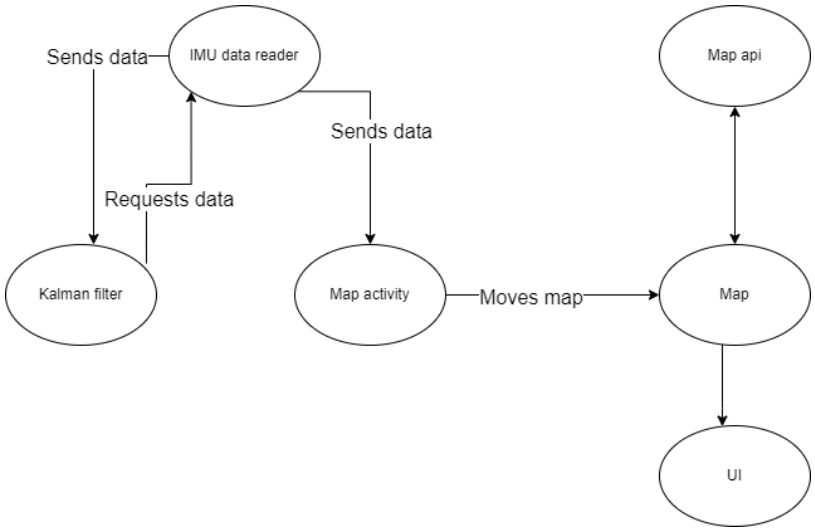
Grafické rozhraní

Při návrhu grafického rozhraní jsem se zaměřil na použitelnost a funkcionalitu předpokládané výsledné aplikace. Z pohledu funkcionality by bylo dobré přidat změnu módu ovládání mapy. To znamená, že výsledná aplikace může obsahovat element pro přepínání mezi módy, který může uživatel využít. Může to být například přepínač nebo kombinace přepínače s tlačítkem, pokud módů bude více než 2. Navíc by bylo dobré mít pro uživatele možnost změny rychlostí posuvu mapy, pokud si uživatel přeje posunout mapu dále při stejném pohybu nebo rotaci mobilního zařízení. Změnu rychlosti by šlo provádět například pomocí dvou tlačítek: zvětšení nebo zmenšení rychlosti. Jiný způsob, jak jde implementovat změnu rychlosti, je horizontální slider. Předpokládaný návrh grafického rozhraní je na obrázku 3.1.



Obrázek 3.1: Grafický návrh aplikace.

3.2 Návrh aplikace



Obrázek 3.2: Schéma výsledného návrhu aplikace.

Aplikace bude rozdělena do několika částí 3.2. Každá část musí mít svůj algoritmus pro správné zobrazení a pohybování mapy při pohybu mobilního zařízení.

Aplikace musí být schopna na základě pohybu zařízení uživatelem získat správná data z kamery a hardwarových jednotek pro odhad změny polohy kamery mapy.

Aplikace je plánována na základě softwarové architektury model-view-controller, která rozděluje program do třech nezávislých komponent: datový model (model), uživatelské rozhraní (view) a řídicí logiku (controller). První komponent definuje data pro použité nástroje,

druhy komponent je zobrazení mapy pro uživatele, třetí komponent se zabývá aplikační logikou aplikace a reaguje na změnu polohy mobilního zařízení.

3.3 Ovládaní pohybu mapy

V této sekci se popisují způsoby ovládání mapy a obecný popis navrhované aplikace. Cílem návrhu je implementovat mapovou aplikaci a propojit mezi sebou pohyb mobilního zařízení s ovládáním mapy.

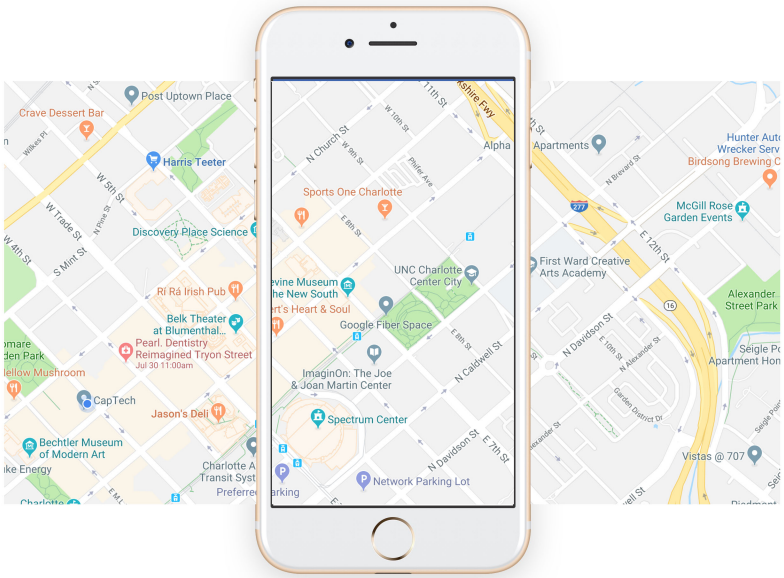
Existuje několik možnosti, jak detekovat změnu polohy chytrého telefonu. Současné chytré telefony nabízí řadu dostupných senzorů, které se používají v různých aplikacích. Navíc se senzory používají v operačních systémech telefonu, například při detekování otáčení mobilu a tím pádem spuštěná aplikace otáčí displej, případně jiné elementy, jako senzorovou klávesnici.

Příkladem, kde se senzory pohybu používají víc jak na detekci otáčení, může být implementace prohlídky panoramatických snímků v mobilní aplikaci Facebook. Ovládání prohlídky takových obrázků pro uživatele je realizováno pomocí rotací nebo translací telefonu. Aplikace Facebook používá data z akcelerometru, která se převádějí na posuv fotografie. Příklad panoramatického snímku je zobrazen na obrázku 3.3.

Obrázek 3.3: Ukázka panoramatického snímku v aplikaci Facebook[1].

[1]Převzato z https://techcrunch.com/wp-content/uploads/2017/08/360-photo-panorama.jpg

Ovládání mapy je naplánované tak, že si uživatel představí, že má mapu přímo na nějaké ploše pod mobilním telefonem a pomocí mobilního zařízení ji může zkoumat. Myšlenka výsledné aplikace je taková, že rotace nebo translace chytrého telefonu způsobí změnu mapy odpovídající prohlídce představené fyzické mapy pod telefonem. Pomocí dostupných technologií lze propojit změnu polohy chytrého telefonu a ovládání pohybem mapy. Ilustrace obecné myšlenky aplikace je na obrázku 3.4.



Obrázek 3.4: Myšlenka výsledné aplikace.

3.4 Mapa

Pro zjednodušení implementace je potřeba rozdělit standardní aplikace mapy na několik části:

- Map activity
- Obecný modul mapy, který komunikuje s API a je zobrazován uživateli

Komponenta „Map activity“ se zabývá nastavením správných koordinátů kamery mapy na základě vstupů zpracovaných dat pohybu. Mapa může, ale nemusí mít zakázaný způsob prohlížení pomoci senzorového displeje. Pohyb mapy musí být realizován na základě aktuálních koordinátů kamery mapy a měnit se se správnými koeficienty vstupních zpracovaných dat pohybu mobilního zařízení. Pro nastavení vhodných koeficientů lze aplikaci testovat na různých uživatelích a měnit podle potřeby urychlení nebo zpomalení pohybu mapy. Komponenta musí pohybovat mapou v 6 směrech: vlevo, vpravo, nahoru, dolu, přiblížení, oddálení.

Komponentu „Map“ lze implementovat na základě již existujících aplikací, které jsou dostupné jako open-source. Příkladem takové aplikace může byt Google Maps nebo Seznam.cz. Sama mapa může obsahovat standardní funkcionalitu jako sledování geopozice

uživatele, hledaní ulic/objektů na mapě, kompas nebo změnu zobrazování mapy (foto ze souputníků/malovaná mapa). Mapa musí čerpat aktuální data (např. změna ulic/objektů na mapě) a být díky tomu vždy aktuální na každý den. Proto lze využít open-source API, ze kterých aplikace bude získávat současná data a tím mapu obnoví. Takovou funkcionalitu poskytuje například společnost Google, která vývojářům dovoluje implementovat vždy aktuální mapu za podmínky, že jejich aplikaci budou využívat připojení na internet. Většinou je potřeba nastavit správný API klíč pro automatickou aktualizaci mapy v aplikací.

Komponenta „Map“ přijímá také data o změně kamery, posílá výsledek uživateli a zobrazuje aplikaci včetně pozice uživatele na mapě.

3.5 IMU reader

Modul je určen pro získání dat z hardwarových jednotek mobilního zařízení. Hlavním cílem této části aplikace je správné zpracování získaných dat ze senzorů pohybu jako akcelerometr, gravitační senzor nebo lineární akcelerometr. Je potřeba získat správná data o změně polohy mobilního zařízení a poslat je do modulu Kalmanova filtru pro další odstranění šumu. Modul musí mít nastavenou správnou frekvencí čtení dat ze senzorů pro optimalizaci výsledné aplikace.

Tento modul se zabývá také výpočtem aktuálních souřadnic, které jsou filtrované pomocí Kalmanova filtru. Kvůli tomu, že data jsou reprezentována zrychlením, je potřeba vypočítat přesný čas, rychlost a pozici mobilního zařízení.

Pří návrhu aplikace je třeba se zaměřit na výběr rychlosti načítání dat. Frekvence načítání dat hraje důležitou roli pro náročnost aplikace na zařízení. Každý senzor povoluje nastavit frekvenci v různých módech. Příklad módů frekvencí v Android Studio:

- `SENSOR_DELAY_NORMAL` – 5 Hz
- `SENSOR_DELAY_UI` – 16,6 Hz
- `SENSOR_DELAY_GAME` – 50 Hz
- `SENSOR_DELAY_FASTEST` – nejrychlejší, zpoždění je 0, záleží na zařízení

Rychlost senzorů hraje velkou roli při výpočtech pozic a Kalmanové filtraci v reálném čase. Na rychlosti závisí výsledná aplikace, konkrétně její reakce na změnu polohy mobilního zařízení. Při nejnižší frekvenci ovládání mapy bude mít přesný posuv se zpožděním. V případě nejvyšší frekvence na různých zařízeních komponenta Kalmanova filtru může provádět filtrací dat se zpožděním. Výběr módu pro načítání dat musí být odůvodněný těmito faktory.

3.6 Kalmanův filtr

Komponenta „Kalmanův filtr“ je implementace matematického Kalmanova filtru. Především je třeba přemýšlet o aktualizaci hodnot zrychlení a pozice, nastavit správný koeficient chyby šumu. Modul přijímá data od komponent „IMU reader“ a „Video processing“, zpracovává je podle algoritmu a odesílá zpět. Kvůli tomu, že data jsou dynamická, komponenta bude aktualizovat matice v momentě, kdy přichází nové hodnoty. Hlavním cílem tohoto Modulu je odstranit šum a tím zlepšit kvalitu pohybu mapy v aplikací. „Kalmanův filtr“

dostává data z „IMU Reader“ ve formátu:

$$\begin{bmatrix} a_x & a_y & a_z & t & x_p & y_p & z_p \end{bmatrix}$$

Kde x je hodnota zrychlení osy x u mobilního zařízení, y je hodnota zrychlení osy y u mobilního zařízení, z je hodnota zrychlení osy z u mobilního zařízení a t je čas, ve kterém hodnoty byly naměřeny. x_p je vypočtená pozice na souřadnici x, y_p je vypočtená pozice na souřadnici y a z_p je vypočtená pozice na souřadnici z.

A vrátí data zpět:

$$\begin{bmatrix} x_{pr} & y_{pr} & z_{pr} \end{bmatrix}$$

Kde x_{pr} je predikce pozice na souřadnici x, y_{pr} je predikce pozice na souřadnici y a z_{pr} je predikce pozice na souřadnice z

Na základě získaných dat Kalmanův filtr musí provést predikci a korekci nových hodnot s odstraněným šumem. Pro lepší odstraněni šumu lze provést řadu experimentů a sestavení hodnot proměnných zodpovědných za chybu výpočtu v Kalmanově filtru.

Pro implementaci tohoto modulu lze využít vzorce 2.5, 2.5 a 2.6.

Kapitola 4

Demonstrační aplikace

Tato kapitola popisuje způsob implementace finální verze mobilní aplikace pro prohlížení mapy metodou pohybu mobilního zařízení. Na začátku jsou popsány vybrané prostředky pro implementaci a architektura výsledné aplikace. Dále jsou popsány implementace jednotlivých komponentů a testování s experimenty.

Pro lepší pochopení problému vývoje dané aplikace jsem se rozhodl udělat řadu experimentů souvisejících s testováním senzorů pohybu na platformě Android. Experimenty jsou popsané v sekci 4.7. Na základě těchto experimentů bylo rozhodnuto, že výsledná aplikace umožní ovládání pohybu mapy pomocí rotací mobilního zařízení. Změnu velikosti mapy (zoom) pomocí senzorů pohybu nelze realizovat. Výsledná aplikace má dva způsoby, jak lze mapu ovládat pomocí rotací chytrého telefonu a „rychlosti“ přemisťování mapy. Implementační detaily těchto způsobu jsou popsaný v sekcích 4.3 a 4.4. První způsob jsem pojmenoval „mode 1“ a z pohledu uživatele je ovládání mapy realizováno tak, že v případě otáčení mobilu, který ve výsledku bude v pozicí X (telefon se nachází pod nějakým stejným úhlem ze všech 3 souřadnic), se vždy zobrazí stejná část mapy za podmínky, že rychlost uživatel nebude měnit. Faktický je to interpretace mapy pod chytrým telefonem, kterou uživatel pozoruje. Druhý způsob ovládání mapy jsem pojmenoval „mode 2“. Tento způsob dovoluje pohybovat mapou na libovolnou vzdálenost. V případě rotací pod nějakým úhlem se mapa začne pohybovat do okamžiku, dokud se uživatel nerozhodné ji zastavit metodou otočení telefonu do pozice, kdy zadní část telefonu nebude kolmo k zemi nebo úhel bude v opačném směru (v takovém případě se mapa zastaví a začne se pohybovat jiným směrem). Zaměřil jsem se na to, aby aplikace byla použitelná, proto existuje i klasický způsob ovládání pohybu mapy – pomocí senzorového displeje.

4.1 Zvolené prostředí

Výslednou aplikaci jsem se rozhodl implementovat v prostředí **Android Studio**. Především proto, že Android Studio má hodně výhod a sama platforma Android umožňuje implementovat své aplikace na různých operačních systémech kromě „IOS“.

Android Studio poskytuje hodně různých nástrojů pro vývoj aplikaci. Například vypisuje log během testování aplikace, má dostupný simulátor mobilního zařízení a zobrazuje statistiku náročnosti na procesor za běhu aplikace. Další výhodou je velmi propracovaný editor kódu, který umožňuje jednoduché odhalení chyb.

Hlavní výhodou Android Studio v tomto zadaní je přístup k hardwarovým jednotkám, což často webové aplikace neumožňují. Aplikace byla implementována v jazycích `Java`

a `Python`, frontend je popsán pomocí klasického způsobu v Android Studiu – popis elementů v `xml`.

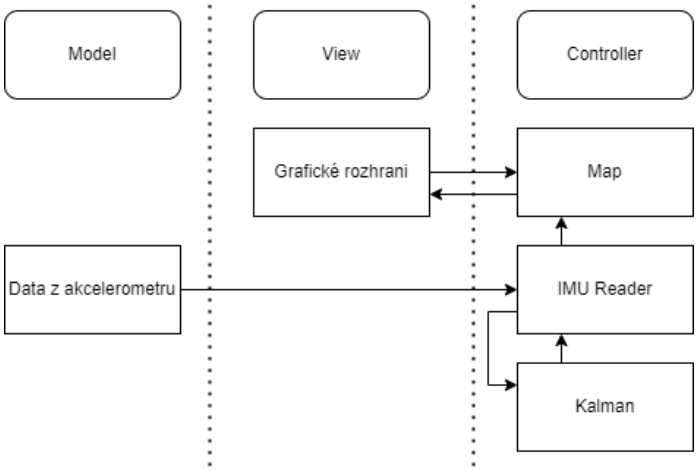
Moje volba padla na Android Studio díky tomu, že Google poskytuje své služby na tomto prostředí, což znamená, že mohu pro vývoj využít knihovny od Google zdarma. Použití služeb Google pro vytváření aplikace, která musí zobrazit mapu, je řešení, které ušetří hodně času.

Pro implementaci Kalmanova filtru a Vizuální odometrii jsem použil jazyk `Python`. V Android Studio jsem použil open-source SDK **Chaquopy**, což umožňuje vytvářet zvláštní třídu s popisem metod a komunikovat s algoritmem napsaným v Javě. Nastavit knihovny pro `Python` lze v souboru `build.gradle` na modulární úrovni.

Jazyk `Python` má své výhody oproti `Javě` a `C++`, které jsou rovněž dostupné v Android Studiu. Rozhodl jsem se použít `Python`, protože je jednoduchý pro matematické výpočty s maticemi.

4.2 Architektura a použití aplikace

Při implementaci jsem zaměřil na přehlednost a znovupoužitelnost kódu. Díky tomuto přístupu aplikace může být v budoucnosti jednoduše rozšířena a předělána. Jde o doplnění funkcionality programu, například přidání nových možnosti mapy pro uživatele. Aplikace je sestavena z několika zvláštních komunikačních bloků, které zpracovávají vstupy a generují výstupy. Základ aplikace je postaven na architekturním vzoru model-view-controller. Logika aplikace je postavená na několika komponentech, což jsou „Imu Reader“, „Kalman Filter“ a „Map“. Podrobnější architektura je popsaná na obrázku 4.1.



Obrázek 4.1: Architektura aplikace.

- **Model** – popisuje data, se kterými aplikace pracuje. Neví nic o stavu ovládacích prvků.
- **View** – reprezentuje uživatelské rozhraní, které je definováno v XML. Popisuje grafické prvky, které uživatel vidí v aplikací.
- **Controller** – popisuje aplikační logiku, proto je nejdůležitější komponentou této architektury. Je komunikační jednotkou mezi View a Modelem. Model je zodpovědný za správu dat aplikace. Důležité je, že uchovává data v takových strukturách, které vyvolávají události, kdykoliv se změní jejich hodnota.

Použití aplikace

Po spuštění aplikace na mobilním zařízení s platformou Android začíná inicializace mapy přes Google API, proto zařízení musí mít připojení na internet a mít povolené použití internetu pro danou aplikaci. Na začátku je nastavený způsob pohybu mapy klasickou metodou přes senzorový displej. Vpravo nahoře je dostupný přepínač mezi módy pohybu mapou (jde o ovládání mapy přes senzorový displej a pohybem mobilního zařízení). V případě, že přepínač je aktivní, mapou lze pohybovat pomocí rotací mobilu, prohlídka pomocí senzorového displeje bude nedostupná. Níže se objeví dvě tlačítka pro přepínání mezi módy pohybu a slider pro nastavení rychlostí posuvu mapy. První režim ovládání mapy umožní pohybovat kamerou, která se dívá na „statickou“ mapu, kterou si uživatel představí pod mobilem. V případě změny rychlosti mapy s ní lze pohybovat o větší vzdálenost se stejným úhlem otáčení mobilu. Pokud se uživatel rozhodné stisknout tlačítko „mode 2“ (výchozí je zapnuté tlačítko s nápisem „mode 1“), změní se mu způsob ovládání mapy. Tento způsob umožní pohybovat mapou s nekonečným pohybem na stranu protilehlou úhlu otáčení. Při změně rychlosti se mapa začne pohybovat rychleji. Příklad grafického rozhraní výsledné aplikace je zobrazen na obrázku 4.4.

4.3 Mapa

Pro návrh mapy jsem použil Google Map SDK pro Android. Pro správnou implementaci je potřeba nastavit API klíč, který aplikaci umožňuje bezpečně odkazovat na mapu. Komponentu lze rozdělit do několika modulů: Map activity, Map a Google Map API.

Pro získání dat aktuální mapy se používá modul „Google Map API“, který je automaticky nastaven v aplikaci a pomocí serveru Google získá nejaktuálnější mapu založenou na serveru Google Maps.

Inicializace mapy probíhá pomocí připojení k internetu přes Google API, proto je důležité mít na mobilním zařízení nastavené Google Services. Pro testování jednotlivých komponent maps bylo zvoleno virtuální zařízení Android, které je k dispozici v Android Studiu.

Google Map SDK poskytuje jednoduchou implementaci základních funkcionalit standardní mapy jako jsou sledování geopozice uživatele, vyhledávání ulic a objektů a řadu dalších nástrojů pro implementaci mapy. Toto je důležitá podmínka výběru open-source softwaru pro splnění požadavků uživatele.

Navíc platforma Google Map SDK podporuje implementaci nejen pro zařízení s operačním systémem Android, ale i řadu dalších platforem jako Web nebo „IOS“. V dnešní se „Google Map SDK“ používá v aplikacích, které potřebují mapy [4].

Všechny elementy související s uživatelským rozhraním jsou propojeny s logikou aplikace ve třídě `Map`. Při inicializaci třídy vytváří objekt každého elementu a nastavuje související

parametry. Například element `SeekBar` má nastavenou minimální a maximální hodnotu koeficientu rychlosti, která odpovídá rychlosti posuvu mapy. Některé elementy jsou na začátku nedostupné pro uživatele, proto jsou automaticky nastaveny neviditelnými. Komponenta `Map` má zvláštní implementaci metod pro každý element GUI a tím pádem zajišťuje komunikaci mezi elementy grafického rozhraní a ostatními komponenty aplikace. Logika programu je zaměřena na změnu odpovídajících proměnných a metod propojených tříd. Například při změně režimu ovládaní mapy pomocí pohybu mobilního zařízení budou vynulovány aktuální pozice mapy a přepnut režim výpočtu polohy metodou `setMode(boolean flag)` objektu `ImuCollector`. Při změně hodnoty grafického elementu `SeekBar` bude hodnota koeficientu rychlosti změněna na novou odpovídající hodnotu.

Pro praktičtější použití jsem se rozhodl přidat přepínač módu, ve kterém uživatel může vybrat způsob posuvu mapy. Základní způsob umožňuje provádět změnu mapy pomocí senzorového displeje, což je mnohem lepší, když někdo potřebuje přemístit mapu na velkou vzdálenost. Druhý způsob reprezentuje mapu pod mobilem a pomocí rotací zařízení se lze na tuto mapu dívat z různých pohledů. Při přepínání způsobu se používají metody třídy `ImuCollector` `pause()` a `resume()`, které odpovídají za čtení dat a výpočty pozic mobilu.

Po inicializaci třídy mapy se vytváří objekt `ImuCollector` a nastaví `Listener` pro reakcí na okamžitou změnu plochy. Poloha se vynásobí hodnotou rychlostí (y musí byt opačná, proto se násobí na zápornou konstantu), která odpovídá hodnotě elementu `SeekBar`. Třída `Map` uchová tyto souřadnice a pak vypočítává změnu pro posuv mapy. Mapa se pohybuje pomocí metody `moveCamera`. Proměnná `isSet` se používá pro přepínání mezi základním režimem ovládání mapy (pomocí senzorového displeje) a režimem pohybu mobilního zařízení. Příklad implementace posuvu mapy je zobrazen ve výpisu 4.1.

```
1       ImuCollector.setListener(new ImuCollector.Listener() {
2           @Override
3           public void onTranslation(float tx, float ty, float tz) {
4
5               if (prev_position != null) {
6                   dx = tx - prev_position[0];
7                   prev_position[0] = tx;
8                   dy = ty - prev_position[1];
9                   prev_position[1] = ty;
10                  dz = tz - prev_position[2];
11                  prev_position[2] = tz;
12
13                  if (imuCollector.getMode()) {
14                      // in 2 mode need to be slower
15                      dx *= speed / 30;
16                      dy *= -speed / 30;
17
18                  } else {
19                      dx *= speed;
20                      dy *= -speed;
21                  }
22
23                  if (isSet) {
24                      map.moveCamera(CameraUpdateFactory.scrollBy(dx, dy));
25                  }
26                  } else {
27                  prev_position = new float[3];
28                  prev_position[0] = tx;
```

```
                prev_position[1] = ty;
                prev_position[2] = tz;
            }
        }
```

Výpis 4.1: Ukázka kódu změny kamery mapy ve třídě `Map`

4.4 IMU Reader

Pro tuto komponentu jsem implementoval třídu `ImuCollector`, která při inicializaci zvolí správná nastavení pro čtení hardwarových jednotek IMU. Pro dosažení cílů je potřeba inicializovat objekt třídy `Sensor`, který bude získávat data z akcelerometru mobilního zařízení. Inicializace senzoru probíhá v metodě `register()`, kde se načtení dat ze senzoru nastavuje na `SENSOR_DELAY_GAME`, což odpovídá frekvenci 50 Hz (perioda je 0,02 *sekundy*). Zpracování dat je implementováno pomoci metody `onSensorChanged(SensorEvent sensorEvent)` dostupné v Android Studio. Pro načtení dat z akcelerometru se používá „low-pass filter“. Princip filtru je postaven na výpočtu gravitační hodnoty zrychlení `ValuesAccelGravity` z dat akcelerometru. Pro výpočet zrychlení, na základě kterého bude vrácená hodnota pozice, se používá rozdíl mezi hodnotami z akcelerometru a gravitačního zrychlení. Princip „low-pass filtru“ je zobrazen ve výpisu 4.2.

```
        for (int i = 0; i < 2; i++) {
                        // sensorEvent.values - raw data from accelerometr
                        valuesAccel[i] = sensorEvent.values[i];
                        valuesAccelGravity[i] = (float) (0.1 * valuesAccel[i] +
                            0.9 * valuesAccelGravity[i]);
                        valuesAccelMotion[i] = valuesAccel[i] -
                            valuesAccelGravity[i];
                    }
```

Výpis 4.2: Ukázka kódu „low-pass filtru“ pro výpočet zrychlení

Třída `ImuCollector` vytváří dva objekty třídy Kalman, vzhledem k tomu, že Kalmanův filtr je implementován v jazyce **Python**, vytváří se instance `Python` pro komunikaci s modulem Python. Každý objekt třídy `Kalman` provádí filtraci jedné ze dvou dostupných souřadnic x a y a na základě zrychlení vrací aktuálně vypočtenou pozicí s odstraněným šumem. Výstupem jsou pozice x a y, které se odesílají pomocí metody `onTranslation()`.

Vzhledem k tomu, že dva dostupné módy ovládání mapy jsou implementovány pomocí čtení dat z akcelerometru, ve třídě `ImuCollector` jsou implementované další metody pro přepínání módu: `resume()`, `pause()` a `setMode(boolean modeType)`. Implementační rozdíl mezi *mode 1* a *mode 2* je postaven na principu výpočtu aktuální pozice. U prvního režimu se používá odhadnutá pozice z Kalmanova filtru, druhý režim počítá součet pozic, což umožňuje pohybovat mapou nekonečno dlouho za podmínky, že telefon je pod stejným úhlem. Příklad výpočtu je ukázán ve výrazu 4.3.

```
        if (!mode) { // First mode
            prediction[0] = kfx.callAttr("update", valuesAccelMotion[0]).toFloat();
            prediction[1] = kfy.callAttr("update", valuesAccelMotion[1]).toFloat();

        } else { // Second mode
```

```
            prediction[0] += kfx.callAttr("update", valuesAccelMotion[0]).toFloat()
                ;
            prediction[1] += kfy.callAttr("update", valuesAccelMotion[1]).toFloat()
                ;
        }
        }
```

Výpis 4.3: Ukázka kódu výpočtu pozic pro každý z režimů

4.5 Kalmanův filtr

Kalmanův filtr je implementován jako 1D filtr, to znamená, že pracuje vždy pouze s jednou souřadnicovou hodnotou. Proto byl pro každý ze dvou směrů vytvořen zvláštní objekt ve třídě `ImuCollector`.

Pro tuto komponentu byla vytvořena třída `KF` v jazyce `Python`, která při inicializací nastavuje proměné `dt` a `Acc_Var` na hodnoty 0,02 (což odpovídá periodě načtení zrychlení z akcelerometru) a 0,5 (což je rozptyl zrychlení). Pak se vypočítají matice `F`, `H`, `Q`, `R`, `X0` a `P0`. Kde `F` je matice přechodu, `H` je matice pozorování, `Q` je kovarianční matice přechodu, `R` je kovarianční matice pozorování, `X0` je inicializační matice stavu a `P0` je kovarianční matice stavu. Inicializace těchto matic je zobrazená ve výpisu 4.4.

```
        # transition_matrix
        F = [[1, dt, 0.5*dt**2],
             [0, 0, dt],
             [0, 0, 1]]

        # observation_matrix
        H = [0, 0, 1]
        # transition_covariance
        Q = np.array(
            [[(dt**4)/4, (dt**3)/2, (dt**2)/2],
             [(dt**3)/2, (dt**2), dt],
             [(dt**2)/2, dt, 1]]
        ) * Acc_Var*1.0 # higher the coeficient, more delayed system
        # observation_covariance
        R = np.array([Acc_Var*10]) # higher number, more smooth and delayed
        # initial_state_mean
        X0 = [0,
              0,
              0]
        # initial_state_covariance
        P0 = [[ 0, 0, 0],
              [ 0, 0, 0],
              [ 0, 0, Acc_Var]]
        }
```

Výpis 4.4: Ukázka kódu výpočtu matic pro Kalmanův filtr

Princip třídy `KF` je implementován pomocí výpočtu odhadnuté pozice. Proto se používá knihovna openCV dostupná v jazyce Python – `pykalman`. Při inicializací objektů `KF` se vytváří objekt z této knihovny a inicializační matice se předávají jako parametry. Konkrétně

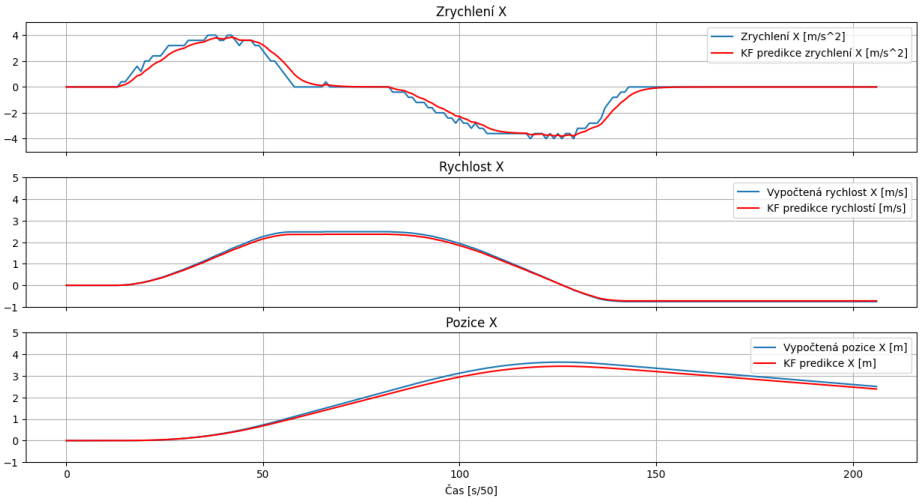
výpočet provádí metoda `filter_update()` ve třídě `KalmanFilter` z knihovny `pykalman`. Výpočet provádí metoda `update()`, která dostává na vstup zrychlení, provede výpočet aktuální pozice a vrátí výsledek.

Funkčnost Kalmanova filtru jsem zkontroloval na hodnotách z akcelerometru. Při experimentu byla vypočítána rychlost a pozice pomoci vzorců 4.1.

$$\begin{aligned} v[t] &= a[t] * dt + v[t-1] \\ s[t] &= v[t] * dt + s[t-1] \end{aligned} \tag{4.1}$$

kde v – rychlost$[m/s]$, a – zrychlení $[m/s^2]$, s – pozice $[m]$, dt = změna času $[s]$.

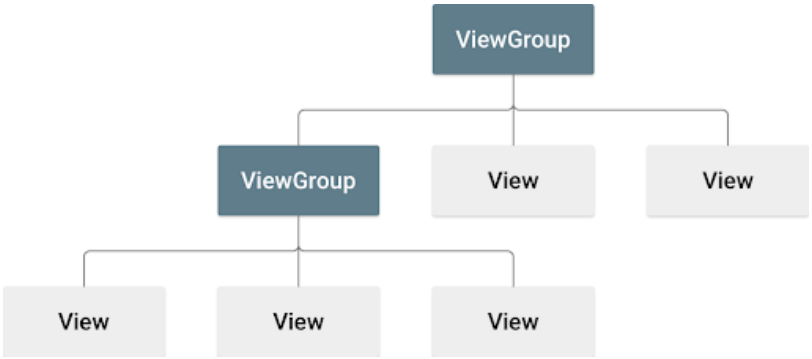
Rychlost a pozice se byly během experimentu vypočítány pro porovnání výsledku. Na výsledném grafu 4.2 je vidět, že Kalmanův filtr odstranil šum u hodnot akcelerací a tím pádem výsledná pozice byla odhadnuta správně a stejně s odstraněným šumem.



Obrázek 4.2: Výsledek odhadu Kalmanova filtru

4.6 Grafické rozhraní

Grafické rozhraní je popsáno pomocí klasické metody v Android Studiu (víz. 4.3 v jazyce `XML`. Rozložení v `XML` definuje strukturu uživatelského rozhraní v aplikaci, například v aktivitě. Všechny prvky v rozložení jsou vytvořeny pomocí hierarchie objektů `View` a `ViewGroup`. Pohled obvykle kreslí něco, co může uživatel vidět a s čím může pracovat. Zatímco `ViewGroup` je neviditelný kontejner, který definuje strukturu rozložení pro `View` a další objekty `ViewGroup`.



Obrázek 4.3: Hierarchie objektů v Android Studio[1].

Objekty `View` se obvykle nazývají „`widgety`“ a mohou být jednou z mnoha podtříd, jako je `Button` nebo `TextView`. Objekty `ViewGroup` se obvykle nazývají „rozložení“ a mohou být jedním z mnoha typů, které poskytují jinou strukturu rozvržení, jako je `LinearLayout` nebo `ConstraintLayout`. „Widgety“ lze jednoduše připojit k aplikační logice díky standardním knihovnám Android Studia. Jakýkoliv „widget“ lze upravovat nebo měnit a kontrolovat jeho stav pomocí objektů v `Javě`, což je velkou výhodou v vývoje dané aplikace.

Ukázka XML kódu v Android Studiu je zobrazená ve výrazu 4.5.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

Výpis 4.5: Ukázka XML kódu v Android Studio

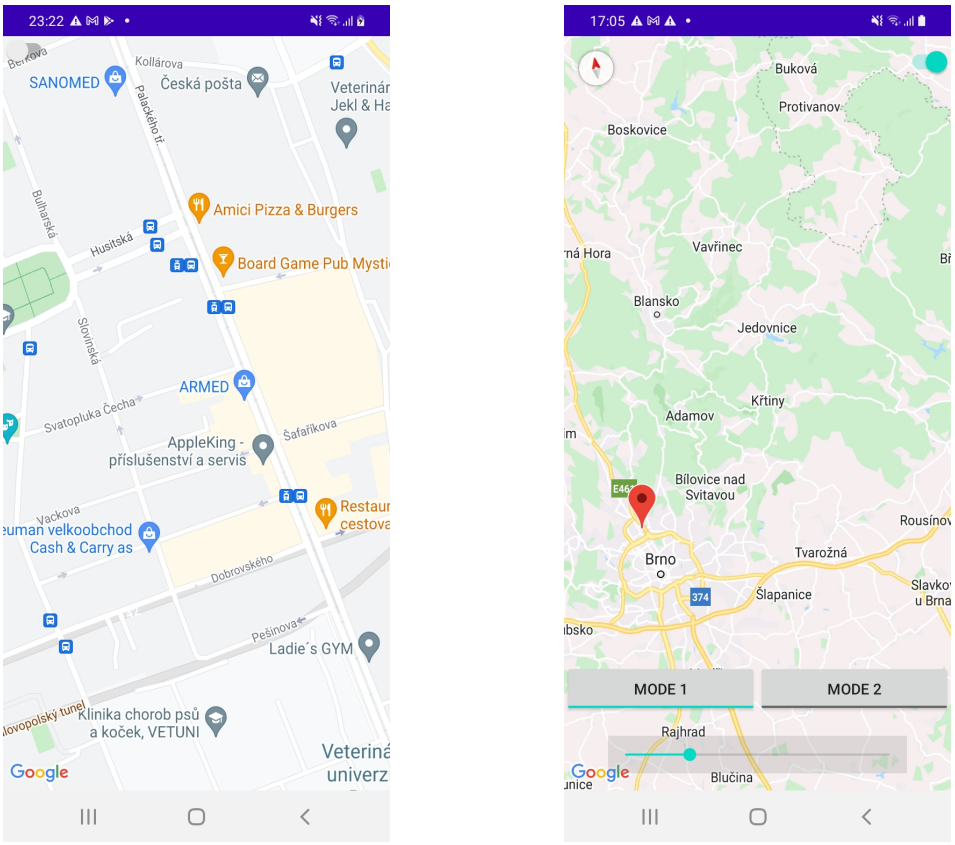
Vzhled aplikace

Aplikace slouží především pro zobrazení mapy. Nejdříve se ukazuje mapa se zarovnáním na Českou Republiku. Uživatelské rozhraní je jednoduché a minimalistické. Hlavním cílem je přepínání mezi módy pohybu mapy, které jsem vyřešil přidáním klasického přepínače typu `Switch` pro zapnutí a vypnutí režimu v Android Studiu, kdy uživatel mapu ovládá pohybem. Výhodou daného elementu je jednoduché pochopení pro uživatele, který mód je nastaven. Když je `Switch` aktivní, je označen modrou barvou, což je pro člověka snadno pochopitelné. V případě, že je `Switch` aktivní, zobrazí dvě tlačítka typu `Button`, která slouží pro přepínání mezi módy ovládání mapy pomocí pohybu mobilního zařízení. Také

[1]Převzato z https://developer.android.com/guide/topics/ui/declaring-layout

navíc zobrazí element `SeekBar`, který označuje rychlost. Uživatel může zvolit zvětšení nebo zmenšení rychlostí posunutím slideru na elementu `SeekBar`.

Mapa má označené země, města a ulice, což je velkou výhodou pro informativní pochopení a orientací. Příklady výsledné aplikace mapy jsou na obrázcích 4.4.



Obrázek 4.4: Výsledná aplikace.

4.7 Testování a experimenty

V předchozích sekcích byla podrobněji popsána implementace celé aplikace po jednotlivých fázích návrhu systému. Program je založený na zpracování dat ze senzorů mobilu při pohybu, proto je nutné provést testování jednotlivých částí za účelem dosažení nejvyšší možné úspěšnosti celého systému. Uživatel očekává správně výsledky a vysokou úspěšnost aplikace. Z tohoto důvodu dobré testování pomáhá dosáhnout nejlepšího možného řešení v každé části programu. V následující sekci je podrobněji popsáné testování a experimenty během vývoje aplikace.

Experimenty se senzory pohybu

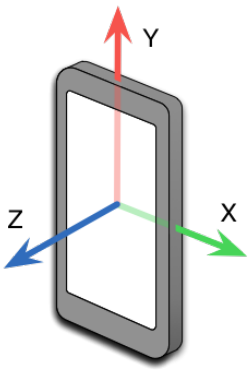
V této sekcí je popsán výběr správného senzoru pro finální implementaci. Během vývoje jsem provedl řadu experimentů a analýzy dát přicházejících ze senzorů mobilního zařízení. Všechny experimenty s načítáním probíhaly na mobilu **Samsung Galaxy A20e** na platformě **Android**. Experimentální kód je implementován v `Android Studiu`. Vstupem programu byla nezpracovaná data z akcelerometru a lineárního akcelerometru, která jsou dostupná na dánem modelu telefonu. Jako výstup byl generován soubor ve formátu CSV s časem a třemi osami.

Cílem těchto experimentů bylo určit způsob implementace výsledné aplikace a vybrat vhodný senzor pro řešení problému. Na základě analýzy dat z různých směrů a způsobu pohybu bylo určeno, jak implementovat program a který senzor použit. Každý experiment se skládá z načtení dat za stejných podmínek z obou senzorů a porovnání výsledků. Díky výsledkům lze pochopit, ve kterých případech který senzor zpracovává pohyb lépe. Je důležité se zaměřit na detekci pohybu pro další vypočtení pozic, které pak způsobí pohyb mapy a současně vyřeší jeden z hlavních problémů zadání. U experimentů se vypočítá pozice ze zrychlení, což pomáhá analyzovat výstupní data z akcelerometru a lineárního akcelerometru.

Data se načítala v režimu `SENSOR_DELAY_NORMAL`, což odpovídá frekvencí 5 Hz, a periodě 0,2 sekundy. Tuto frekvenci jsem zvolil, protože byla vhodná pro porovnání dat a analýzu. Při načtení dat senzory neměly zpoždění a čas mezí výstupy senzorů byl konstantní.

Bylo rozhodnuto provést dva experimenty z důvodu lepšího pochopení problematiky výběru správného senzoru pohybu na platformě Android. První experiment předpokládá, že při obecném návrhu aplikace a řešení hlavního problému uživatel bude pohybovat mapou pomocí translací telefonu. Druhý experiment je potřebný pro řešení daného problému pomocí jiného přístupu k posunu mapy. Tento přístup je postavený na principu pohybu mapy pomocí rotací mobilního zařízení kolem souřadnic X a Y.

Pro lepší pochopení experimentů, které jsou popsány na principu souřadnic mobilního zařízení, je nutno rozumět, jak jsou souřadnice reprezentované vzhledem k telefonu na obrázku 4.5.



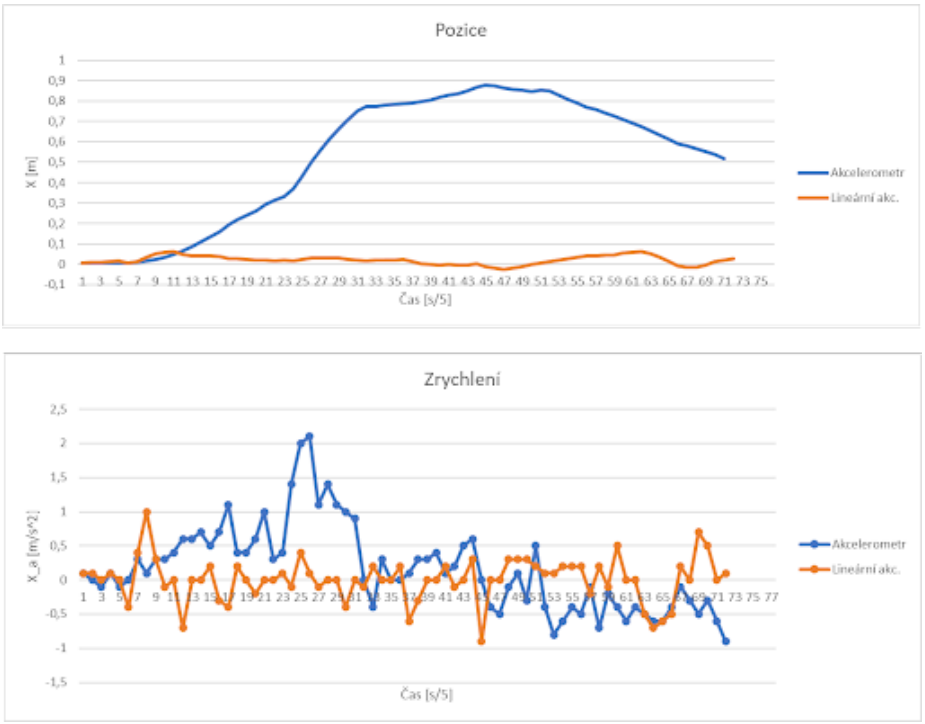
Obrázek 4.5: Souřadnice na chytrém telefonu[2].

Experiment s translací mobilního zařízení ve směru souřadnice X

Popis experimentu: načtení dat ze senzorů při pohybu mobilního zařízení vlevo a potom vpravo na stejnou pozici (ve směru souřadnice x). Mobilní zařízení je v poloze, kde souřadnice Z je kolmá k zemi. V konečné fázi experimentu telefon musí být v počáteční pozici.

[2]Převzato z https://medium.com/@pivithuruamarasinghe/android-accelerometer-reorientation-c1d3867aa15b

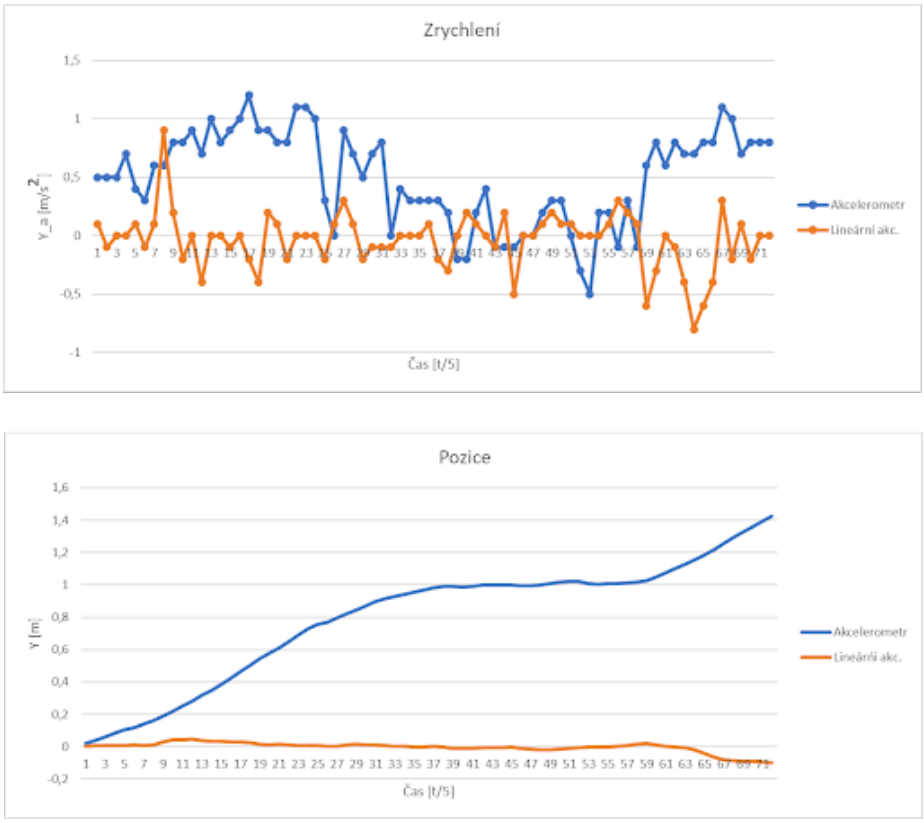
Grafy ze souřadnice X:



Obrázek 4.6: Výsledek experimentu č. 1, souřadnice X.

Analýza experimentu: Na druhém grafu na obrázku 4.6 je vidět, že akcelerometr vrací hodnoty téměř odpovídající reálné situací. Současně vypočtená pozice zhruba odpovídá reálné poloze mobilu. Na druhou stranu, když byl mobil posunout zpět na počáteční pozici, poslední hodnota pozice je $\sim 0{,}52\ m$, i když má být $\sim 0\ m$. Oproti tomu lineární akcelerometr vrací chaotické hodnoty a během experimentu se vypočtená pozice skoro vůbec nemění.

Grafy ze souřadnice Y :



Obrázek 4.7: Výsledek experimentu č. 1, souřadnice Y.

Analýza experimentu: Předpokládá se, že pozice na souřadnici Y se v takovémto experimentů nemusí skoro vůbec měnit. Akcelerometr naměřil data, která se na první pohled mohou zdát jako šum ze senzorů, ale reálně se vrátí velká změna pozice v kladném směru na souřadnici Y, což neodpovídá reálnému pohybu mobilního zařízení. Lineární akcelerometr tady fungoval lépe, protože data zrychlení jsou pouze šumem a pozice opravdu odpovídá hodnotě ~ 0, která je správná.

Grafy ze souřadnicí Z:

Obrázek 4.8: Výsledek experimentu č. 1, souřadnice Z.

Analýza experimentu: Stejně jako u souřadnice Y se pozice na ose Z nemusí měnit. Na grafu se zrychlením je dobře vidět, že hodnoty z akcelerometru jsou přibližně konstantní a odpovídají $\sim 9{,}8\ m/s^2$. Je to způsobeno tím, že akcelerometr počítá gravitační sílu země. Mobil se nacházel v pozici, kde souřadnice Z je kolmá na povrch, proto pozice, která se zde vypočítala stejně jako na osách X a Y, velmi rychle roste a neodpovídá realitě. Lineární akcelerometr vrací správně zrychlení bez ohledu na šum během experimentu a proto můžeme pozorovat, že výsledná pozice je vypočtená skoro správně bez ohledu na šum.

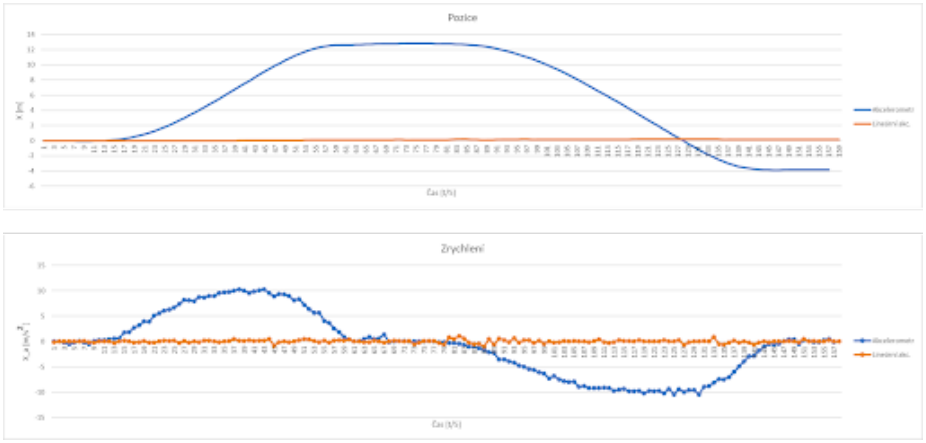
Směrodatná odchylka zrychlení:

	Akcelerometr	Lineární akcelerometr
x	0,62363	0,31585
y	0,40017	0,24131
z	0,51946	0,45090

Experiment s rotací mobilního zařízení kolem souřadnice X

Popis experimentu: načtení dat ze senzorů při rotaci mobilního zařízení vlevo a vpravo ve směru souřadnice X. Mobilní zařízení je na začátku v poloze, kde souřadnice Z je kolmá k povrchu. V konečné fázi experimentu telefon musí být v počáteční pozici.

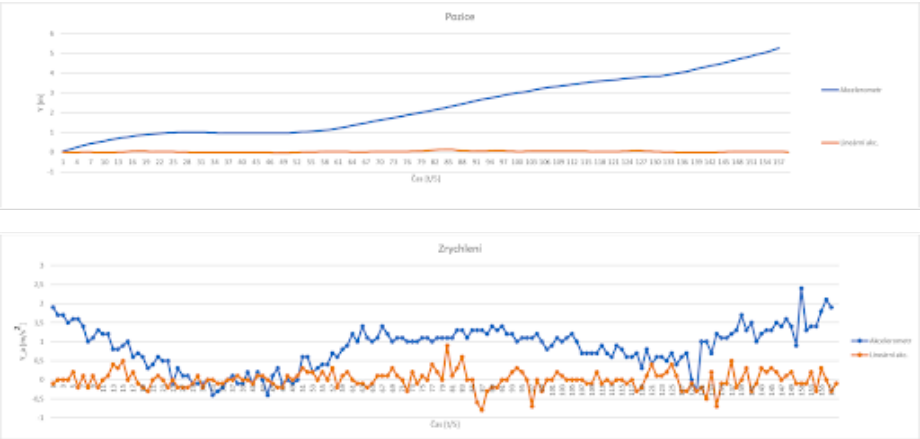
Grafy se souřadnicí X:



Obrázek 4.9: Výsledek experimentu č. 2, souřadnice X.

Analýza experimentu: Na grafu zrychlení data z akcelerometru na souřadnici X nabývají kladných hodnot při rotaci mobilu vlevo a záporných při rotaci vpravo. Na konci experimentu se veličiny přibližují k počáteční hodnotě 0. Chyba je způsobená nepřesným úhlem, který byl při experimentu připuštěn v konečné pozici (osa Z musí být kolmá k zemi). Je dobře vidět, že pozice celkem odpovídá správné reálné poloze mobilního zařízení během rotací a je to tedy dobrý výsledek. Oproti tomu lineární akcelerometr na souřadnici X vrací během celého experimentu skoro nulové hodnoty a vypočtená pozice se také nemění.

Grafy se souřadnicí Y :



Obrázek 4.10: Výsledek experimentu č. 2, souřadnice Y.

Analýza experimentu: Akcelerometr vrací malé hodnoty ze souřadnice Y. Střední hodnota zrychlení je 0,845 m/s^2, což způsobil náklon mobilního zařízení s malým úhlem ve směru osy Y za běhu experimentu. Je to důvod, proč se pozice ve směru osy Y měnila během času, když měla zůstávat konstantní kolem 0. Lineární akcelerometr stejně jako u osy X nevracel hodnoty, které by detekovaly pohyb telefonu, proto pozice zůstala stejná.

Grafy ze souřadnicí Z:

Obrázek 4.11: Výsledek experimentu č. 2, souřadnice Z.

Analýza experimentu: U souřadnice Z akcelerometr začal klesat a potom růst, což odpovídá realitě otáčení mobilního telefonu. Vzhledem k tomu, že mobil se nacházel na začátku v pozici, kde souřadnice Z je kolmá k zemi, počáteční hodnoty byly kolem tíhového zrychlení. V momentě otáčení o 90 stupňů zrychlení nabývá hodnot blízkých k 0 a při vrácení do počáteční polohy roste k číslu tíhového zrychlení. Vypočtená pozice na souřadnici Z klesá v momentech, kdy se telefon nenachází v poloze kolmé na osu X, jinak je konstantní. přitom lineární akcelerometr vrátí hodnoty skoro jako v předchozích případech daného experimentu.

Směrodatná odchylka zrychlení:

	Akcelerometr	Lineární akcelerometr
x	6,14177	0,29246
y	0,54964	0,23047
z	3,42759	0,37795

Vyhodnocení experimentů

V prvním experimentu se podařilo vyčíst skoro správnou polohu mobilního zařízení na souřadnici X pomocí akcelerometru. Velký problém působí nesprávná finální pozice na ose X, což v implementaci reálného řešení nebude pohybovat mapou správně. Když například uživatel posune telefon jedním směrem a pak zpět, dostane mapu na jiné pozici, než byla na začátku. A při velmi malé rychlosti se mapa pravděpodobně pohybovat nebude. Lineární akcelerometr neukazuje zrychlení správně v žádném z experimentů, proto ho nelze použít. Většina dat přicházejících z lineárního akcelerometru na mobilním zařízení je prostě šum, proto jej bohužel nelze použít pro implementaci. V druhém experimentu akcelerometr vrátil data, ze kterých se podařilo vypočítat reálnou pozici mobilního zařízení. Problém je, že ze souřadnice Z nelze žádným způsobem získat správná data a pozici. Proto nebude možné realizovat přiblížení a oddálení mapy. Experimenty taky ukazují, že senzory pohybu neodstraňují šum, takže je nutné zlepšit výsledné řešení pomocí filtru.

Uživatelské testování

Testování na uživatelích je důležitou částí vývoje softwaru. Vzhledem k tomu, že mobilní aplikace, které je věnovaná táto práce, nemá jiných podobných programu, bylo rozhodnuto provést sadu testovacích experimentů s různými uživateli. Testování probíhalo na mobilním zařízení **Samsung Galaxy A20e**, aplikace byla nainstalována pomocí Android Studio.

Na začátku každému uživatelovi byla představena aplikace. Každý uživatel věděl základní principy ovládání mapou v různých módech. V testování zúčastnilo 10 uživatelů, kteří dostali následující úlohy a byli pozorování při jejich vyplnění.

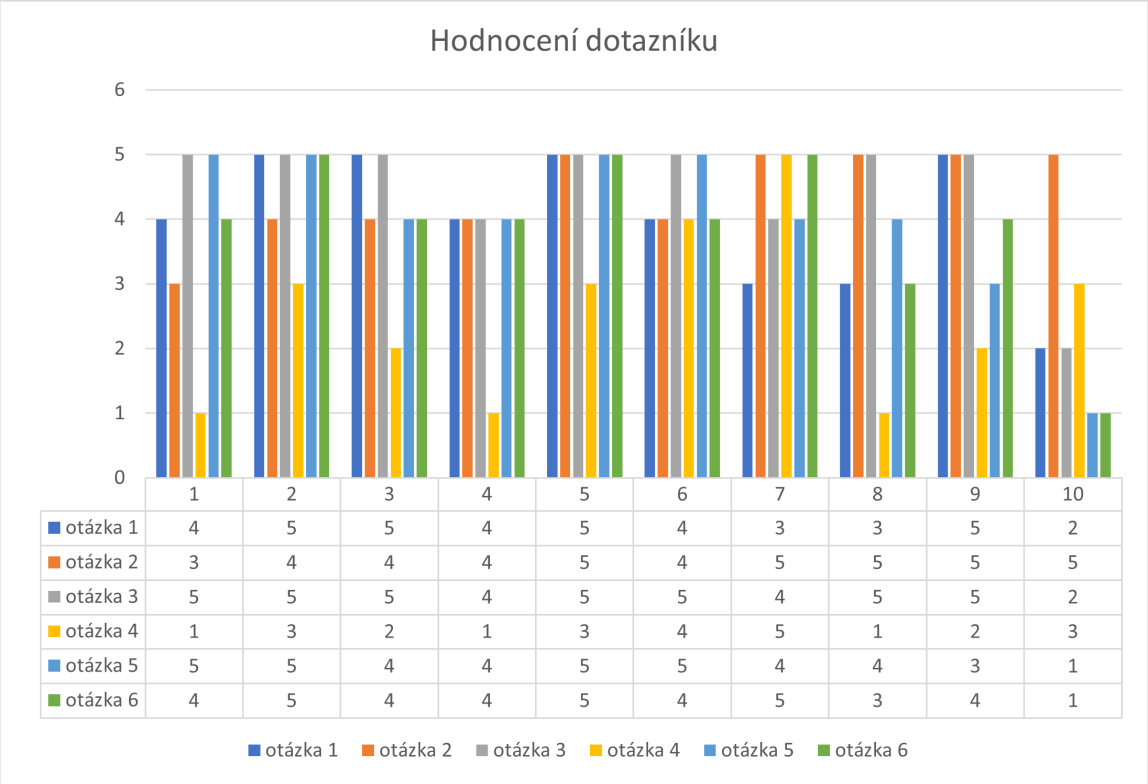
1. Najít na mapě Brno v režimu, který je dostupný při inicializací aplikace.
2. Přiblížit mapu na Fakultu Informačních Technologií VUT, tak, aby bylo vidět křižovatku vedle pošty na Mojmírovo náměstí a park „Božetěchova“.
3. Zapnout pomocí přepínače režim ovládání mapou pomoci pohybu telefonu.
4. V režimu “mode 1” a pomocí změny rychlosti přemístit mapu ve stranu tramvajové zastávky „Semilasso“ a pak ve stranu „Královo Pole“ nádraží.
5. Přepnout se do „mode 2“ a oddálit mapu pomocí senzorového displeje.
6. Přemístit mapu v režimu „mode 2“ ve stranu Prahu.
7. Pomocí senzorového displeje přiblížit mapu tak, aby bylo vidět „Karlův“ most.

Během těchto úloh bylo prováděno pozorování na uživatelích se zaměřením na srozumitelnost použití aplikace. Byla sledovaná taky komunikace uživatele s grafickým návrhem, konkrétně pochopitelnost přepínání mezi módy a zvetšení nebo zmenšení rychlosti pohybu mapy. Po testování každý uživatel vyplnil dotazník B s ohledem na uživatelské rozhraní a pochopitelnost použití aplikace. Výsledky dotazníku jsou na obrazku 4.12.

Výsledky testování

Testování ukázalo, ve kterém režimu mapou bylo ovládat mapou pochopitelně a kde uživatel měl problém. Taky bylo zjištěno, kde uživatel strávil nejvíce času. Největší problémy byly v úlohách číslo 6 a 7. Tyto problémy konkrétně ukázaly, že nevšechní uživateli rozumí tomu, jak pohybovat mapou v druhém režimu. Problém je v tom, že při náklonu mobilu, mapa se pohybuje protilehlým směrem oproti úhlu náklonu. Uživatelské rozhraní nedělalo skoro žádný problém pro přepínání mezi způsoby ovládání mapou. Další úlohy ukázali pozitivní výsledek testování.

	Předpokládaný čas	Výsledek
Splnění úloh 1-4	60 s	66 s
Splnění úloh 5-7	80 s	78 s
Celkem	140 s	144 s



Obrázek 4.12: Výsledky dotazníku.

Kapitola 5

Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat mobilní aplikaci pro ovládání mapou pomocí pohybu mobilního zařízení. Hlavním zadáním bylo detekovat pohyb chytrého telefonu a interpretovat to v možnost ovládat mapou pro uživatele.

Na začátku jsem prostudoval metodu Vizuální odometrie pro jednu kameru, dostupné inerciální měřící jednotky na mobilním zařízení, Kalmanův filtr a dostupné technologie a metody postupu pro vývoj mobilních aplikací. Pro implementaci jsem se rozhodl použít senzory pohybu s odstraněním šumu a vhodné technologie pro vývoj svojí mobilní aplikace.

Návrh aplikace vycházel z definice problematiky dáne aplikace a potřeb uživatelů. Na základě tohoto byla zvolena návrhová architektura a návrh uživatelského rozhraní. Funkcionalita každé komponenty byla popsaná v návrhu pro další vývoj.

Zvolil jsem implementovat výslednou aplikaci v prostředku Android Studio, což byla moje první zkušenost ve vývoji mobilních aplikací na platformě Android. Celá aplikace je implementovaná v programovacích jazycích Java a Python. Během implementace jsem se setkal s některými problémy. Největším problémem bylo nesprávné načtení dat z lineárního akcelerometru. Tento problém jsem vyřešil změnou způsobu ovládání mapy. Nakonec jsem provedl testování výsledné aplikace na uživatelích. Ve výsledcích se ukázaly správný návrh uživatelského rozhraní a pochopitelnost ovládání mapou pomocí pohybu smartphonu.

Zlepšit budoucí aplikaci je možné pomocí přidání modulu, který bude zpracovávat video z kamery, pro implementaci nového způsobu ovládání mapou pomoci translace telefonu.

Výsledkem této bakalářské práce je funkční aplikace, která je spustitelná na reálném zařízení.

Literatura

[1] Becker, A. *KalmanFilter* [online]. [cit. 2022-05-02]. Dostupné z: https://www.kalmanfilter.net/default.aspx.

[2] D. Nister, O. N. a Bergen, J. *Visual odometry.* Washington, DC, USA: IEEE, 2004. ISBN 0-7695-2158-4.

[3] FISCHLER, M. A. *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography.* Commun, 1981. ISBN 0001-0782, 381-395.

[4] Francis, N. *Collection of the Coolest Uses of the Google Maps API* [online]. [cit. 2022-05-02]. Dostupné z: https://www.jotform.com/blog/collection-of-the-coolest-uses-of-the-google-maps-api/.

[5] Hassenzahl, M. a Tractinsky, N. User experience - a research agenda. *Behaviour & Information Technology.* Taylor & Francis. 2006, sv. 25, č. 2, s. 91–97. DOI: 10.1080/01449290500330331. Dostupné z: https://doi.org/10.1080/01449290500330331.

[6] Kim, Y. a Bang, H. Introduction to Kalman Filter and Its Applications. In: Govaers, F., ed. *Introduction and Implementations of the Kalman Filter.* Rijeka: IntechOpen, 2018, kap. 2. DOI: 10.5772/intechopen.80600. Dostupné z: https://doi.org/10.5772/intechopen.80600.

[7] Kultová, A. *UX design* [online]. [cit. 2022-05-04]. Dostupné z: https://wiki.knihovna.cz/index.php/UX_design.

[8] Lepetit, V., Moreno Noguer, F. a Fua, P. EPnP: An accurate O(n) solution to the PnP problem. *International Journal of Computer Vision.* Únor 2009, sv. 81. DOI: 10.1007/s11263-008-0152-6.

[9] Nister, D. *An efficient solution to the five-point relative pose problem.* IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. ISBN 0-7695-1900-8.

[10] Paul, Z. *Howard Musoff. Fundamentals of Kalman Filtering: A Practical Approach.* American Institute of Aeronautics and Astronautics, Incorporated, 2000. ISBN 978-1-56347-455-2.

[11] Rahul Sukthankar, R. G. S. a Mullin, M. D. *Smarter Presentations: Exploiting Homography in Camera-Projector Systems* [online]. 2001 [cit. 2022-05-02]. Dostupné z: http://www.cs.cmu.edu/~rahuls/pub/iccv2001-rahuls.pdf.

[12] Wikipedia contributors. *Inertial measurement unit — Wikipedia, The Free Encyclopedia*. 2022. [Online; accessed 9-May-2022]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Inertial_measurement_unit&oldid=1083981921.

[13] Wikipedia contributors. *Kalman filter — Wikipedia, The Free Encyclopedia* [https://en.wikipedia.org/w/index.php?title=Kalman_filter&oldid=1086395690]. 2022. [Online; accessed 9-May-2022].

[14] Yousif, K., Bab Hadiashar, A. a Hoseinnezhad, R. An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics. *Intelligent Industrial Systems*. Listopad 2015, sv. 1, s. 10. DOI: 10.1007/s40903-015-0032-7.

Příloha A

Obsah DVD

Přiložené DVD obsahuje:

- zdrojové kódy aplikace
- soubory pro instalaci aplikace na mobilní telefon
- zdrojové soubory textu bakalářské práce v systému LaTeX
- soubor REARME.md - manuál k instalací
- experimenty
- prezentační video
- plakát
- text bakalářské práce ve formátu PDF

Příloha B

Dotazník k testování

1. Jak byste zhodnotili obtížnost uživatelského rozhraní?
   Těžko 1 2 3 4 5 Lehko
2. Jak byste zhodnotili obtížnost úloh?
   Těžko 1 2 3 4 5 Lehko
3. Jak byste zhodnotili ovládání mapy v režimu „mode 1“?
   Nejhorší 1 2 3 4 5 Nejlepší
4. Jak byste zhodnotili ovládání mapy v režimu „mode 2“?
   Nejhorší 1 2 3 4 5 Nejlepší
5. Jak byste zhodnotili celkovou orientací v aplikaci?
   Nejhorší 1 2 3 4 5 Nejlepší
6. Obecné hodnocení aplikace
   Nejhorší 1 2 3 4 5 Nejlepší
7. Co byste v aplikaci zlepšili?

Příloha C

Plakát

Obrázek C.1: Plakát