Univerzita Pavla Jozefa Šafárika v Košiciach

Prírodovedecká fakulta

LOKALIZÁCIA V INDOOR PROSTREDÍ S VYUŽITÍM AKCELEROMETRA A KOMPASU

DIPLOMOVÁ PRÁCA

Študijný odbor:	Informatika
Školiace pracovisko:	Ústav informatiky
Vedúci záverečnej práce:	RNDr. František Galčík, PhD.

Košice 2015

Bc. Miroslav Opiela

Poďakovanie

Moje poďakovanie patrí predovšetkým vedúcemu tejto diplomovej práce, ktorým bol RNDr. František Galčík, PhD. Ďakujem mu za jeho ochotu a čas, cenné rady a nápady.

Abstrakt

Určenie aktuálnej pozície používateľa je kľúčovou otázkou, ktorú musí riešiť akákoľvek navigačná aplikácia. Kým v typických navigačných riešeniach sa využíva GPS, v budovách nie je možné pracovať s týmto satelitným signálom. Rôzne prístupy k indoor lokalizácii sú každoročne prezentované na konferencii IPIN [1]. V našej práci využívame na určenie aktuálnej polohy používateľa senzory, ktoré sú prítomné v bežnom smartfóne – akcelerometer a kompas. Pozorovaním zmien hodnôt akcelerometra sa detekuje vykonaný krok. Kompas slúži na určenie smeru chôdze. Spoločne poskytujú možnosť určenia relatívnej pozície vzhľadom na danú navigačnú trasu [2]. V tejto práci prezentujeme vlastný algoritmus, ktorý je schopný lokalizovať používateľa aj bez známej navigačnej trasy. Tento prístup sa využíva v oblasti robotiky [3]. Mapu budovy reprezentujeme pomocou mriežky s dostatočne malým rozmerom jedného políčka. Pre každé takéto miesto v budove počítame pravdepodobnosť, s akou sa tam používateľ nachádza. V práci opisujeme aj efektívnu implementáciu nášho algoritmu. Tento prístup poskytuje potenciál pre kombináciu s inými vstupnými informáciami o aktuálnej polohe, ako je WiFi fingerprinting alebo QR kódy. Takto navrhnutý algoritmus je schopný vysporiadať sa s rôznou dĺžkou kroku, nepresnosťami hodnôt z kompasu alebo nesprávne detekovanými krokmi. Zároveň algoritmus využíva štruktúru budovy so svojími chodbami a ich križovatkami k zlepšeniu presnosti lokalizácie. Vytvorený prototyp ukazuje, že algoritmus je funkčný a použiteľný v aplikácii indoor navigácie.

Abstract

Localization of a user is the key element in every application of navigation. GPS is used in typical outdoor navigational solutions, but satelite signal is not available in buildings. There are various approaches presented on IPIN conference [1]. In our solution, we use sensors - accelerometer and compass, available in common smartphones to detect the current position of a user. A step is detected by observation of values of the accelerometer. The compass is used to determine a walking direction. Indeed, it is possible to determine a relative position along the existing navigation path [2]. In the thesis, we describe our own algorithm, which is able to localize a user without any existing navigation path. This approach is used in robotics [3]. A map is reprezented by a grid with small enough size of each field. The probability of user position is computed for all fields in the grid. The thesis descibes the efficient implementation of our algorithm. There is an option for adding more information about the current position from QR codes or WiFi fingerprinting. This algorithm is able to cope with various step lengths, inaccuracy of compass values or wrong detected steps. The algorithm uses the building structure with corridors and their intersections to improve the accuracy of estimation of user position. Created prototype shows that the algorithm is correct and usable in an application of indoor navigation.

Obsah

Úvod **7**

1 Indoor navigácia **9**

1.1 Požiadavky na navigačný systém 9

1.2 Špecifiká indoor navigácie 11

1.3 Existujúce navigačné aplikácie 11

2 Indoor lokalizácia **13**

2.1 Prehľad existujúcich prístupov k indoor lokalizácii 13

2.2 Smartfóny a ich senzory 15

2.3 Metódy určenia absolútnej pozície 16

3 Lokalizácia s využitím akcelerometra a kompasu **18**

3.1 Lokalizácia s explicitne danou navigačnou cestou 18

3.2 Detekcia krokov 19

3.3 Natočenie zariadenia 22

4 Lokalizácia s využitím akcelerometra a kompasu bez explicitne danej navigačnej cesty **24**

4.1 Bayesovské filtre 24

4.2 Lokalizačný pravdepodobnostný model 26

5 Efektívna implementácia **34**

5.1 Implementačné detaily 34

5.2 Lokalizačný pravdepodobnostný model z pohľadu implementácie 35

5.3 Generovanie a uchovávanie máp 38

5.4 Časová a pamäťová zložitosť 42

6 Evaluácia riešenia **45**

6.1 Pomocné nástroje na testovanie 45

6.2 Pozorovania a presnosť lokalizácie 48

Záver **50**

Prílohy **54**

Úvod

Navigačné aplikácie sú v dnešnej dobe bežne dostupné, používané a sú dostatočne presné. Kľúčovou otázkou každej takejto aplikácie je lokalizácia používateľa. Pri bežne používaných navigačných systémoch hovoríme o GPS a satelitnom signáli. Indoor prostredie má svoje špecifiká. V prvom rade ide o neprítomnosť satelitného signálu. Ten nie je dostatočne silný, aby prenikol stenami budov. Kým samotné navigovanie a poskytnutie informácii o navigačnej trase sa nemusí líšiť od outdoor navigácie, tak určovanie polohy si vyžaduje iné prístupy. Lokalizácia v indoor prostredí je tiež viac citlivá na presnosť. Kým chybné určenie pozície o 10 metrov pri cestovaní autom si používateľ nemusí ani všimnúť, pri indoor lokalizácii ide o výrazný problém.

V našej práci identifikujeme špecifiká indoor prostredia a prinášame prehľad rôznych prístupov najmä k otázke lokalizácie v indoor prostredí. Tieto prístupy a výsledky sú prezentované na konferencii *IPIN (International Conference on Indoor Positioning and Indoor Navigation)* [1], ktorá sa koná pravidelne od roku 2010. Vymenuvávame niekoľko riešení z rôznych oblastí prístupu k tomuto problému. Niektoré su založené na simulovaní satelitného signálu, iné napríklad na pozorovaní okolia pomocou kamery alebo iných senzorov. Práca sa zameriava na využitie smartfónov pri určení pozície používateľa, konkrétne na detekciu krokov pomocou senzorov, ktoré sú prítomné v bežnom smartfóne - akcelerometer a kompas.

Hlavnou časťou tejto práce je predstavenie vlastného algoritmu lokalizácie, ktorý je odpoveďou na výzvu určenia pozície používateľa bez explicitne danej navigačnej cesty. Mapa budovy je reprezentovaná pomocou mriežky s dostatočne malým rozmerom jedného políčka. Pre každé políčko sa počíta pravdepodobnosť, že sa tam nachádza používateľ. Opisujeme aj efektívnu implementáciu tohto algoritmu, kedže potrebujeme mať na zreteli obmedzený výpočtový výkon bežného smartfónu. Prinášame porovnanie pôvodného návrhu algoritmu a jeho efektívnejšej implementácie z pohľadu času výpočtu aj pamäťových nárokov. Opisujeme aj spôsob vytvárania a uchovávania informácii, ktoré popisujú mapu budovy.

Algoritmy sú v práci podrobené evaluácii a sú identifikované silné stránky aj slabiny lokalizačného algoritmu. Počas práce bolo vytvorených niekoľko pomocných aplikácii a tiež finálny prototyp indoor lokalizácie.

Ciele práce vieme zhrnúť do týchto bodov:

1. Preskúmať a analyzovať existujúce prístupy k indoor navigácii a lokalizácii s využitím smartfónov.

2. Preskúmať a prakticky overiť možnosti indoor lokalizácie s využitím akcelerometra a kompasu bez explicitne danej navigačnej cesty.

3. Navrhnuté algoritmy implementovať vo forme funkčného prototypu pre platformu Android a analyzovať ich praktickú použiteľnosť (pamäťové nároky, realizovateľnosť výpočtov v reálnom čase na zariadeniach s obmedzenými výpočtovými možnosťami).

Kapitola 1

Indoor navigácia

1.1 Požiadavky na navigačný systém

Každý navigačný systém, a teda aj navigačný systém v indoor prostredí, musí spĺňať nasledovné požiadavky:

Lokalizácia používateľa

Správne určenie aktuálnej pozície používateľa je kľúčovou otázkou každej navigácie. Kým v outdoor prostredí sa využíva satelitný signál, indoor navigácie pri lokalizácii používateľa sú nútené použiť iné riešenie. Dôvodom je, že tento signál nie je schopný prejsť cez steny budov. Dôležítým aspektom je presnosť lokalizácie. Pri ceste autom môže byť tolerovaná chybná detekcia pozície aj o niekoľko desiatok metrov. Pri indoor lokalizácii je vyžadovaná oveľa väčšia presnosť, niekedy vyjadrená až v centimetroch.

Zvolenie cieľa

Navigácia musí poskytnúť možnosť zvoliť si cieľ cesty. Aktuálnu pozíciu môžeme považovať za začiatok cesty. Pri indoor navigácii je potrebné myslieť na fakt, že daná budova nemusí mať iba jedno poschodie. Tomu treba prispôsobiť aj výslednú navigačnú aplikáciu.

Nájdenie trasy

Nie vždy musí ísť o najkratšiu cestu. V indoor prostredí sa môžeme stretnúť napríklad so situáciou, že schodisko je bližšie k aktuálnej pozícii ako výťahy. V takom prípade môže byť pre používateľa lepšou cestou tá dlhšia.

Výpočty v reálnom čase

Navigačný systém musí byť schopný reagovať na aktuálny pohyb používateľa.

V tejto práci sa zameriavame na lokalizáciu používateľa. Všetky algoritmy, pomocné nástroje aj prototyp sú navrhnuté tak, aby bolo možné doplniť ďalšie prvky indoor navigácie. V našom prístupe je cieľom ponúknuť všeobecné riešenie, ktoré nebude fungovať iba pre jednu konkrétnu budovu, ale poskytne algoritmy použiteľné v univerzálnej aplikácii indoor navigácie prístupnej bežným používateľom. Definovali sme preto následné doplňujúce požiadavky na navigačný systém:

Absencia prídavných zariadení v budove

Existujú rôzne zariadenia, napr. pseudolity [4], ktoré simulujú satelitný signál. Inštalácia týchto zariadení v budove predstavuje finančnú záťaž, ale hlavne môže byť nevhodné umiestňovať takého zariadenia v niektorých typoch budov, napr. v nemocniciach alebo historických budovách. Pri absencii akýchkoľvek prídavných zariadení poskytneme správcovi budovy ľahkú nasaditeľnosť navigácie pre jeho budovu. Potrebné bude iba vytvoriť mapu so všetkými nevyhnutnými informáciami. V práci spomíname QR kódy, ktoré vyžadujú ich umiestnenie v budove. Neide však o finančne náročnú vec a tieto kódy by nemali prekážať v žiadnom type budovy. Ak by bolo potrebné, tak je možné ich nahradiť inou metódou lokalizácie.

Absencia prídavných zariadení používateľa

Chceme využiť to, že používateľ bude mať aplikáciu navigácie na svojom smartfóne. V dnešnej dobe sú takéto mobilné zariadenia veľmi rozšírené, preto bez akýchkoľvek iných zariadení bude navigačný systém prístupný veľkému počtu používateľov. Potrebujeme využiť iba to, čo bežne dostupný smartfón ponúka. V našom prípade to znamená využitie senzorov.

Univerzálnosť a jednoduché vytváranie máp

Navigačný systém by mal byť funkčný pre akúkoľvek budovu, nie len pre jednu konkrétnu. S tým je spojená aj požiadavka jednoduchého vytvárania máp. Je potrebné si uvedomiť rozdiely medzi budovami. Rozdiely medzi nákupným centrom, letiskovou halou, školou a administratívnou budovou môžu byť v počte poschodí (resp. či ide o jednoposchodovú budovu), v štruktúre (jedna chodba a po stranách miestnosti - kancelárie, učebne alebo väčší otvorený priestor - napríklad letisková hala) a v ďalších aspektoch.

1.2 Špecifiká indoor navigácie

Vo všeobecnosti je najväčší rozdiel medzi outdoor a indoor navigáciou v lokalizácii používateľa. V outdoor prostredí je zisťovaná poloha zariadenia pomocou GPS a satelitného signálu. Tento spôsob je rozšírený, dostupný a dostatočne presný. Satelitný signál však nie je možné zachytiť v budovách tak, aby sa mohla využiť lokalizácia pomocou GPS. Táto skutočnosť ponúka možnosť využitia rôznych prístupov k indoor navigácii. Najnovšie metódy a výsledky sú prezentované na konferencii *IPIN (International Conference on Indoor Positioning and Indoor Navigation)* [1], ktorá sa koná každoročne od roku 2010.

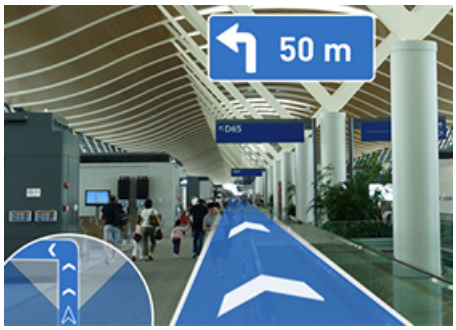
Zaujímavou výzvou, ktorú pri outdoor navigácii nie je potrebné brať do úvahy, sú viacposchodové budovy. Pozícia používateľa, resp. zariadenia nemusí byť jednoznačne určená len súradnicami zemepisnej šírky a dĺžky, ale aj údajom o poschodí, na ktorom sa nachádza. S tým je spojený aj prechod medzi poschodiami pomocou výťahu alebo schodiska. Tento aspekt má dopad na vytváranie máp a zobrazenie používateľovi.

Ďalšou skutočnosťou, ktorá môže mať vplyv na aplikáciu navigácie v budovách je ich rôzna štruktúra. Používateľ môže mať rôzne požiadavky a priority v budovách s otvoreným priestranstvom ako sú letiskové haly alebo veľtrhy a v budovách s mnohými chodbami a miestnosťami ako sú nemocnice alebo úradné budovy.

1.3 Existujúce navigačné aplikácie

Existuje niekoľko komerčných aplikácii, ktoré poskytujú navigáciu v indoor prostredí. Väčšina týchto aplikácii využíva na lokalizáciu prídavné zariadenia v budove alebo kombináciu viacerých prístupov. Kedže naša práca má za cieľ zamerať sa práve na určenie aktuálnej pozície používateľa, tieto metódy sú bližšie opísané v ďalších kapitolách.

Úspešná indoor navigačná aplikácia je aplikáciou na smartfón alebo tablet. Na rozdiel od outdoor prostredia, kde je bežné mať špeciálne zariadenie na navigáciu. Spôsob zobrazenia naviačnej cesty sa nelíši od outdoor navigácie. Táto cesta je väčšinou zobrazená graficky. Niektoré aplikácie [5] ponúkajú aj možnosť rozšírenej reality, kde obraz z kamery je prekrytý doplňujúcou vrstvou s navigačnou trasou a informáciami o vzdialenostiach ako to vidieť na obrázku 1.1.



Obr. 1.1: Rozšírená realita - obraz z kamery doplnený o navigačnú trasu a informácie o vzdialenostiach. Ukážka z aplikácie Infsoft [5].

Tieto komerčné riešenia sú využívané napríklad v letiskových budovách alebo nákupných centrách. Na mape sú priamo zobrazené aj obchody alebo iné miesta záujmu, čo poskytuje používateľovi prehľad o svojom okolí. Existujú tiež nástroje na vytvorenie vlastnej aplikácie indoor navigácie pre svoju konkrétnu budovu [6].

Kapitola 2

Indoor lokalizácia

2.1 Prehľad existujúcich prístupov k indoor lokalizácii

Prezentovaných bolo mnoho prístupov k lokalizácii v indoor prostredí s rôznym potenciálom pre praktické použitie. Ponúkame prehľad a kategorizáciu týchto riešení bez ohľadu na zariadenie, ktoré je lokalizované. Niektoré prístupy sa snažia využiť princíp bežnej outdoor navigácie s využitím GPS. Existujú však aj výskumy, ktoré používajú nové metódy v tejto problematike, napr. oblasť počítačového videnia. Ďalším rozdielom je predmet skúmania. Kým viaceré prístupy skúmajú iba základnú lokalizáciu, iné sa zameriavajú aj na niektoré špeciálne prípady, napríklad smartfón umiestnený vo vrecku nohavíc [7], chôdzu po shodoch [8] alebo navigáciu pre používateľov na invalidnom vozíku [9].

Vysielanie a spracovanie signálu

Je niekoľko riešení [4, 10, 11], ktoré využívajú zariadenia nazývajúce sa pseudolity (pseudo-satelity). Tieto zariadenia simulujú satelitný signál a musia byť strategicky rozmiestnené v budove (prevažne na konci chodieb). Tento prístup k indoor navigácii je veľmi podobný bežnej outoor navigácii. Signál je dokonca možné zachytiť rovnakými zariadeniami. Takéto riešenia nemusia byť smerované len na používanie smartfónov, ale môžu vyžadovať špeciálne zariadenia aj na strane používateľa.

Okrem pseudolitov existujú aj iné zariadenia, ktoré musia byť rozmiestnené v budove a poskytujú možnosť na podobný prístup. Komerčné riešenie *LabWerk* [12] využíva iBeacon zariadenia založené na technológii bluetooth. Tento prístup má výhodu

v tom, že bluetooth zaťažuje batériu smartfónu menej ako GPS.

Ďalší z prístupov je založený na WiFi technológii. Výhodou je, že nie je potrebné budovať žiadnu infraštruktúru v budove iba kvôli lokalizačnému systému. Na lokalizáciu sa využíva fingerprinting metóda založená na sile WiFi signálu a porovnávaním s databázou meraní [13, 14].

Sledovanie okolia

V tomto prístupe sa očakáva prítomnosť mobilného zariadenia so svojimi senzormi, kamerou alebo fotoaparátom. Ide predovšetkým o oblasť počítačového videnia. Spoločnou charakteristikou všetkých riešení využívajúcich tento princíp je hľadanie zhody medzi snímaným okolím a záznamom o budove. Existuje viacero riešení, ktoré majú rôzny predmet snímania.

Môžu ním byť špeciálne značky umiestnené v budove [15], ktoré pripomínajú QR kódy. Mobilné zariadenie neustále sníma okolie a hľadá dané značky, pomocou ktorých určuje aktuálnu pozíciu v budove.

Zaujímavým prístupom je aj lokalizácia, ktorá využíva svetelný senzor zariadenia [16]. Jej základom sú svetlá umiestnené v budove a znalosť ich pozície. Čím viac svetiel v budove je vypnutých, tým väčšia je nepresnosť lokalizácie.

Niektoré riešenia využívajú kombináciu viacerých prístupov. Algoritmy počítačového videnia snímajú nohy používateľa a detekujú urobené kroky [17] alebo snímajú okolie a hľadajú zhodu medzi obrázkami budovy z databázy a obrazom snímaným kamerou [18].

Analýza chôdze

Existuje veľké množstvo riešení s týmto prístupom. Chôdza je analyzovaná pomocou senzorov. Jedným z použiteľných zariadení je *IMU (Intertial Measurement Unit)*, ktoré sa využíva aj v letectve alebo námorníctve. Toto zariadenie obsahuje akcelerometer a gyroskop alebo v niektorých prípadoch magnetometer. Pomocou *IMU* je možné detekovať kroky a použitím ďalších algoritmov lokalizovať používateľa v indoor prostredí [19].

Mnoho riešení používa na detekciu krokov smartfóny, ktoré tiež obsahujú senzory ako sú akcelerometer, kompas alebo gyroskop. Kľúčové úlohy pri tomto prístupe sú detekcie krokov, určenie natočenia používateľa (smer jeho chôdze), odhad dĺžky kroku a samotné určenie pozície. Typickým problémom je nepresnosť samotných dát zo

senzorov, ale aj odhadov dĺžky kroku. Chyba sa môže kumulovať a úspech celého riešenia závisí od schopnosti vysporiadať sa s týmito nepresnosťami.

2.2 Smartfóny a ich senzory

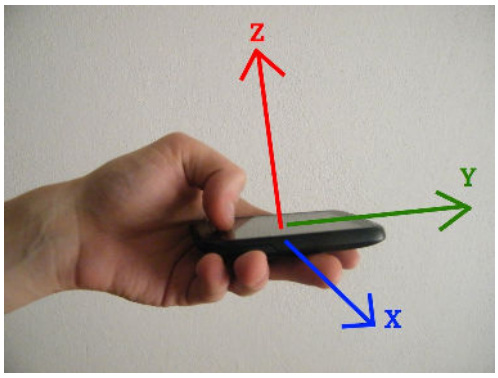
Bežne dostupné smartfóny obsahujú niekoľko senzorov. Platforma Android poskytuje jednoduchý prístup k dátam z týchto senzorov [20]. Rozdeľuje senzory na hardvérové a softvérové. Hardvérové senzory sú fyzicky prítomné na zariadení. Softvérové senzory (nazývané tiež virtuálne) odvádzajú svoje hodnoty z jedného alebo viacerých hardvérových senzorov. Všetky takéto senzory môžeme rozdeliť na tri kategórie:

- *Pohybové senzory* snímajú pohyb zariadenia. Konkrétne pohybové senzory sú akcelerometer, gravitačný senzor, gyroskop, lineárny akcelerometer a rotačný vektorový senzor.
- *Polohové senzory* poskytujú informáciu o polohe zariadenia. Ide o senzor magnetického poľa (magnetometer), senzor otočenia (kompas) a senzor merajúci vzdialenosť od iného objektu.
- *Senzory prostredia* poskytujú údaje o rôznych charakteristikách prostredia. Patrí tu senzor vonkajšej teploty, osvetlenosti, vonkajšieho tlaku vzduchu, vonkajšej relatívnej vlhkosti aj teploty zariadenia.

Mnohé z týchto senzorov majú nulový alebo len veľmi malý potenciál využitia pre lokalizáciu zariadenia. Niektoré senzory sú síce použiteľné pri indoor lokalizácii, ale nevyskytujú sa v každom zariadení alebo sa ich implementácia líši. To napríklad znamená, že lineárny akcelerometer môže byť v jednom zariadení hardvérový senzor a v inom softvérový. Pre našu prácu sú najdôležitejšie tieto dva senzory, ktoré sú hardverové a sú prítomné vo väčšine zariadení:

- *Akcelerometer* je pohybový senzor, ktorý sníma zrýchlenie zariadenia v troch zložkách (obrázok 2.2). Hodnoty z akcelerometra sa použijú pri algoritmoch na detekciu krokov. Pri typickom držaní zariadenia je relevantná vertikálna zložka z.
- *Kompas* je polohový senzor, ktorý sa v kontexte indoor navigácie využíva na určenie smeru chôdze. Pri lokalizácii s využitím detekcie krokov sa kumuluje chyba, s ktorou je potrebné sa vysporiadať. Pri detekcii jedného kroku môže byť až štyridsaťstupňový rozdiel medzi nameraným a reálnym natočením [21].

Podľa výskumu z projektu HiMLoc [8] môže byť táto chyba až 100 stupňov, avšak pri normálnych podmienkach (chôdza nie tesne pozdĺž stien alebo popri kovových štruktúrach) je chyba tolerovateľná.



Obr. 2.2: Tri zložky akcelerometra.

Využitie smartfónov je nevyhnutné pre vytvorenie úspešnej aplikácie indoor navigácie adresovanej širokému okruhu používateľov. Od algoritmu indoor lokalizácie sa vyžaduje, aby bol schopný vysporiadať sa s nepresnosťami zo senzorov. Je potrebné myslieť aj na obmedzený výpočtový výkon bežného smartfónu.

2.3 Metódy určenia absolútnej pozície

Samotný akcelerometer a kompas nie sú schopné určiť absolútnu pozíciu v budove. Pomocou týchto senzorov je možné detekovať kroky a teda určiť zmenu pozície vzhľadom k nejakej počiatočnej pozícii. Pre určenie absolútnej pozície je potrebné použiť inú metódu. Algoritmus použitý na lokalizovanie používateľa, čiže určenie počiatočného miesta je samostaným algoritmom a je možné ho zameniť za iný. Pri demonštrácii našich algoritmov v našom prototype používame QR kódy.

QR kódy

QR kódy sú bežnému používateľovi smartfónov známe. Vieme ich preto použiť na lokalizáciu. Táto metóda je alternatívou k jednoduchému manuálnemu výberu pozície. Pre používateľa je rýchlejšie a pohodlnejšie naskenovať kód ako hľadať v nejakom zozname alebo niečo písať. Táto metóda je stopercentne presná, kedže každý QR kód umiestnený v budove je jenoznačný. Je na poskytovateľovi aplikácie v danej budove, aby zvážil rozmiestenenie týchto kódov v budove. Kódy by mali byť pri všetkých

vstupoch do budovy a na niektorých strategických miestach, kde je pravdepodobnosť, že používateľ zapne aplikáciu navigácie. Ide napríklad o východy z obchodov v nákupnom centre alebo dvere ordinácie lekára v nemocnici. Nevýhodou týchto kódov môže byť potreba skenovania používateľom. Rovnako je potrebné pre poskytovateľa aplikácie v budove vyrobiť QR kódy a rozmiestniť ich v budove.

Obr. 2.3: Ukážka QR kódu, ktorý má informáciu o názve aktuálnej mapy a pozíciu používateľa v centimetroch vzhľadom k tejto mape.

WiFi fingerprinting

Prístup určujúci pozíciu na základe sily signálu WiFi siete je prezentovaný v bakalárskej práci [13] inšpirovanej podobným výskumom [14].

Existuje viacero spôsobov ako určiť pozíciu pomocou sily signálu. Môže ísť o geometrické metódy ako sú trilaterácia alebo triangulácia. V týchto projektoch sa využíva fingerprinting. Táto metóda sa snaží vytvoriť určitú charakteristiku konkrétneho miesta vzhľadom na silu signálu prijímanú z rôznych WiFi vysielačov. V ideálnom prípade sú tieto charakteristiky jedinečné pre každú pozíciu. V prvom rade je potrebné vytvoriť charakteristiky pre rôzne miesta v budove a uložiť ich do databázy. Následne sa pri lokalizácii vykoná meranie a určí sa daná pozícia.

Tento prístup sa ukázal ako dostatočne spoľahlivý ako opisujú experimenty v spomínaných výskumoch. Experimenty boli robené v budovách s mnohými miestnosťami, nie v otvorenom priestore ako je letisková hala. V budove s miestnosťami svoju úlohu zohrávajú steny, ktoré tlmia WiFi signál a tým zlepšujú presnosť určenia zhody fingerprintov. Testovaním sa ukázalo, že je najlepšie urobiť štyri merania. Pochopiteľne s každým ďalším meraním sa zväčšuje presnosť, ale po štyroch meraniach sa táto presnosť už nezvyšuje tak výrazne. Na merania je potrebný istý čas. Aj keď ide len o zopár sekúnd, toto riešenie je vhodné skôr na občasné určenie aktuálnej absolútnej pozície. Táto metóda môže fungovať aj automaticky bez nutnosti dodatočného vstupu od používateľa.

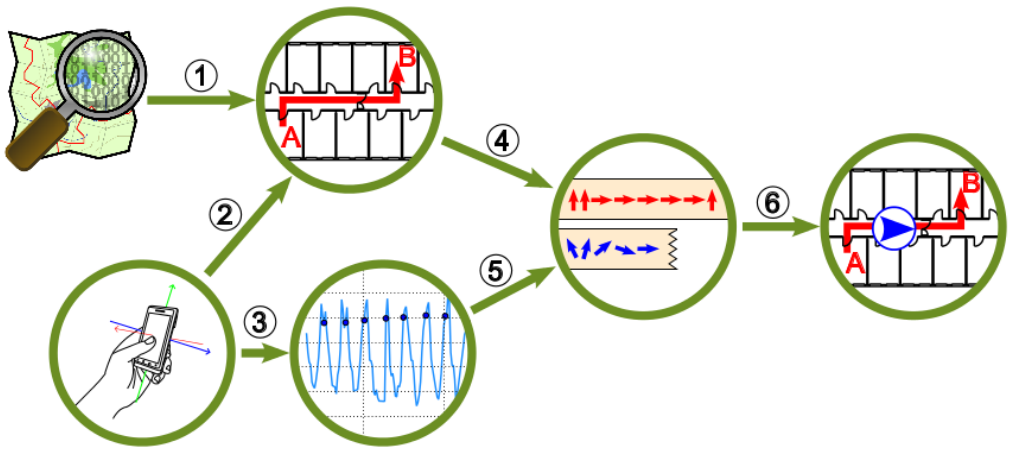
Kapitola 3

Lokalizácia s využitím akcelerometra a kompasu

Kľúčovým prvkom riešení využívajúcich akcelerometer a kompas je detekcia krokov. Určenie pozície používateľa sa počíta vzhľadom k zadanej navigačnej ceste [2, 22]. Tento prístup predpokladá, že používateľ naozaj kráča po danej trase.

3.1 Lokalizácia s explicitne danou navigačnou cestou

Projekt FootPath [2] prináša riešenie s využitím senzorov dostupných v bežnom smartfóne. Pracuje sa s hodnotami z akcelerometra, ktoré poskytujú vstup pre algoritmus detekcie krokov. Pomocou kompasu sa určuje aktuálne natočenie zariadenia. V tomto prístupe sa určuje relatívna pozícia vzhľadom na zadanú trasu. Popis riešenia v niekoľkých krokoch ilustruje nasledovný obrázok 3.4.



Obr. 3.4: Schéma fungovania aplikácie navigácie projektu FootPath

Tento prístup rieši nasledovné udalosti:

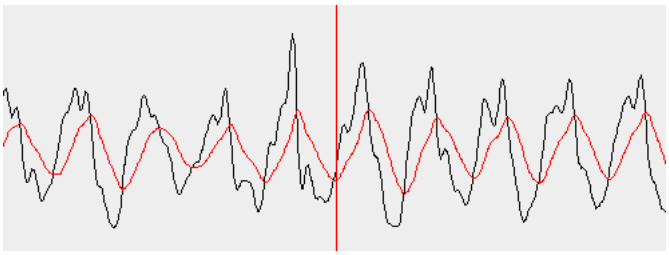
1. FootPath aplikácia získa mapu budovy využitím OpenStreetMap [23].
2. Používateľ zadá aktuálnu a cieľovú pozíciu, čo v konkrétnej aplikácii znamená čísla dverí, pri ktorých sa nachádza a ktoré sú jeho cieľovou pozíciou. Aplikácia vyráta cestu z počiatočnej aktuálnej pozície k cieľu.
3. Aplikácia berie hodnoty z akcelerometra a kompasu a detekuje kroky.
4. Vypočítaná cesta je reprezentovaná sekvenciou odhadovaných krokov. Každý krok je určený svojím natočením.
5. Detekované kroky sú trasformované na podobnú sekvenciu ako pri zadanej trase. Na určenie natočenia sa využijú hodnoty z kompasu.
6. Použije sa algoritmus na zistenie aktuálnej pozície vzhľadom na zadanú trasu. Aplikácia poskytne používateľovi informáciu o aktuálnej polohe.

Dĺžka kroku bola v tomto projekte vypočítaná na základe zadanej výšky používateľa. Aj keď sa ukázala táto metóda ako nepresná, algoritmus na určenie pozície sa vedel vysporiadať s touto nepresnosťou. Na samotné určenie pozície, čiže na hľadanie zhody medzi sekvenciami krokov extrahovaných zo zadanej trasy a detekovanými krokmi, boli použité dva prístupy. Algoritmus, ktorý dosahoval lepšie výsledky, využíva princíp dynamického programovania.

V bakalárskej práci [22] boli testované tieto algoritmy a pridané vlastné vylepšenia. Pri testoch samotnej lokalizácie používateľa sa ukázalo, že výsledok bol závislý od dĺžky kroku. Upravil sa preto algoritmus tak, aby si s týmto aspektom poradil.

3.2 Detekcia krokov

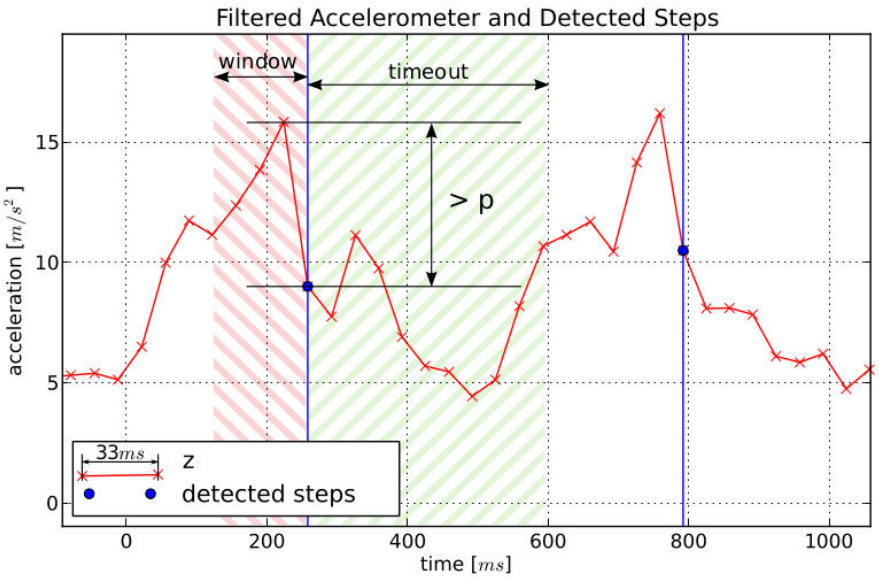
Na detekciu krokov sa využívajú hodnoty z akcelerometra. Aj v algoritme z projektu FootPath, aj v spomínanej bakalárskej práce sa sledujú hodnoty iba z jednej zložky akcelerometra, konkrétne zložky z, ktorá je kolmá na displej zariadenia. Kedže hodnoty získané z akcelerometra obsahujú šum, pri oboch prístupoch sa aplikuje low pass filter (obrázok č. 3.5). Tento filter vyhladí získané dáta. Stráca sa však údaj o konkrétnej hodnote, čo znemožňuje matematické výpočty dĺžky kroku alebo rýchlosti chôdze. Znamená to, že úlohou týchto algoritmov je iba rozhodnúť, či bol urobený krok.



Obr. 3.5: Hodnoty z akcelerometra vyznačené čiernou farbou. Hodnoty po aplikovaní low pass filtra sú vyznačené červenou farbou.

FootPath algoritmus

Algoritmus na detekciu krokov v tomto projekte využíva snímanie zmien hodnôt akcelerometra. Ako vidíme na obrázku č. 3.6, krok je detekovaný, keď sa hodnota akcelerometra zmení o viac ako hodnota p, pričom empiricky bola zvolená hodnota $2m/s^{-2}$. Táto zmena sa sníma v poli *window*. Dôležitou súčasťou je aj *timeout*. V rámci tohto úseku nemôžu byť detekované dva kroky. Ide o relevantnú podmienku, lebo človek nevie urobiť dva kroky v rozmedzí $333ms$, čo je nimi definovaná hodnota *timeout*.

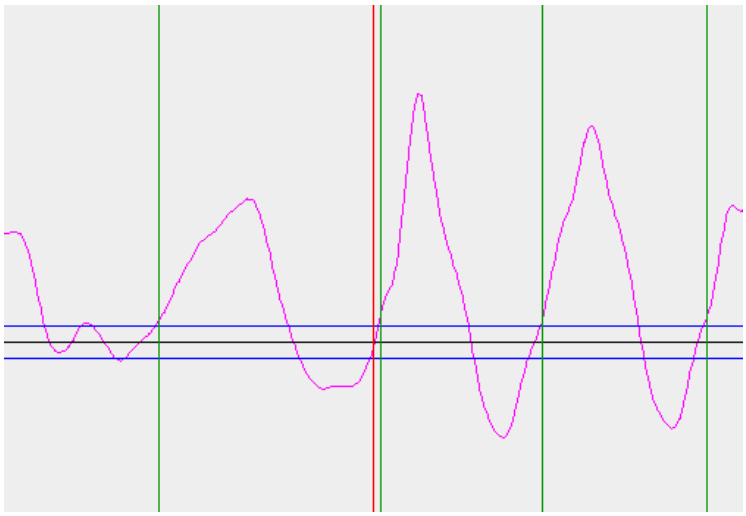


Obr. 3.6: Detekcia krokov algoritmom popísaným v projekte FootPath

Tento algoritmus ukázal dobrú schopnosť vysporiadať sa s rôznou rýchlosťou pohybu, rôznou dĺžkou kroku, rôznou trajektóriou a pohybom po schodoch. Zle detekované kroky boli prevažne na začiatku a na konci chôdze a tiež pri veľmi výrazných zmenách hodnôt akcelerometra.

Štvorfázový algoritmus

Algoritmus vychádza z pozorovania, že hodnoty akcelerometra sa pri každom kroku správajú podobne aj napriek nepresnostiam senzora. Vieme v týchto dátach identifikovať štyri fázy, ktoré sú prítomné, keď je vykonaný krok. Kým je zariadenie v pokoji, akcelerometer uvádza určitú pokojovú hodnotu. Následne pri vykonaní kroku sa táto hodnota zvyšuje, až kým dosiahne lokálne maximum. Potom klesá naspäť na pokojovú hodnotu a pokračuje ďalej na lokálne minimum. V poslednej fáze kroku sa hodnota vráti na pôvodnú hodnotu. Keďže dáta z akcelerometra aj po aplikovaní low pass filtra nie sú ideálne, pokojovú hodnotu sme nahradili intervalom, čo vidieť aj na obrázku č. 3.7.



Obr. 3.7: Detekcia krokov s využitím štyroch fáz pri každom kroku.

Tento algoritmus robil tým menej chybných detekcií, čím väčšie boli kroky alebo čím rýchlejšie boli kroky robené. Výrazne lepšie boli výsledky pri častejších zmenách zo státia na chôdzu a naopak. Horšie výsledky dosiahol v situáciach, keď používateľ menil naklonenie zariadenia, alebo robil pomalé kroky.

Porovnanie algoritmov

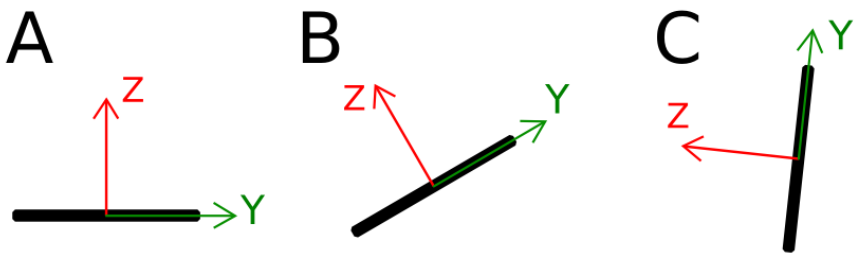
Oba algoritmy dosiahli vysokú spoľahlivosť pri bežnej chôdzi. Rozdiel je ale v druhu chýb, ktoré sa vyskytovali. Algoritmus z projektu *FootPath* je liberálny, detekoval okrem vykonaných krokov aj falošné kroky, ktoré neboli vykonané. Dokázal však správne identifikovať všetky naozaj urobené kroky. Priemerne detekoval štyri falošné kroky na sto dobrých. Náš algoritmus je konzervatívny. Nedetekoval žiaden falošný krok, ale

občas niektorý vykonaný krok nezaznamenal. Pri bežnej chôdzi išlo približne o jeden nedetekovaný krok na sto správych krokov.

Tieto dva prístupom rôzne, ale spoľahlivosťou podobné algoritmy sú dobrým potenciálom pre ďalšie lokalizačné algoritmy. Vieme si podľa potreby zvoliť prístup, s ktorým bude algoritmus lokalizácie dosahovať lepšie výsledky. Cieľom ďalších lokalizačných algoritmov bude okrem iného aj vysporiadanie sa s chybami spôsobenými nepresnými meraniami senzorov a kumulovanými chybami z algoritmu detekcie krokov.

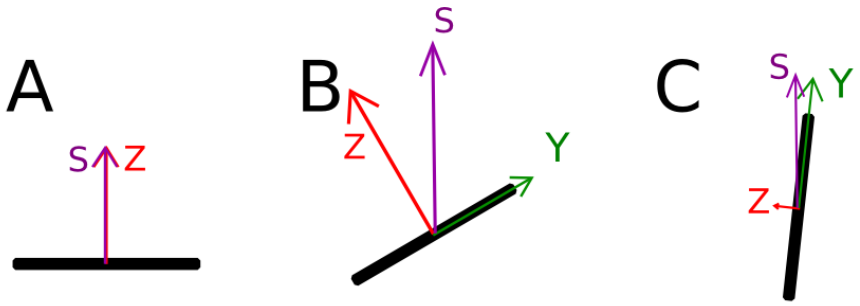
3.3 Natočenie zariadenia

Rôzne natočenie zariadenia spôsobovalo problémy pri oboch algoritmoch na detekovanie krokov. Predpokladalo sa vodorovné držanie zariadenia (obrázok č. 3.8 - situácia *A*). Pri kráčaní sa zariadenie pohybovalo nahor a nadol, algoritmus na detekciu krokov bral do úvahy len jednu zložku akcelerometra (zložka z - na obrázku vyznačená červenou farbou). Pri našom testovaní použiteľnosti algoritmov používatelia nedostali inštrukcie ako držať zariadenie. Ukázalo sa, že aj keď väčšina držala smartfón vodorovne, tak sa našli používatelia, ktorí držali zariadenie zvislo (uhol, ktorý zvieral smartfón s podlahou sa blížil 90°) alebo šikmo (uhol bol niekde v intervale od 30° do 45°). Pri nakloneniach *B* a *C* štvorfázový algoritmus, ktorý vo všeobecnosti dosahoval trochu lepšie výsledky, nebol vôbec schopný detekovať kroky. Algoritmus z projektu FootPath sa ako tak vedel vysporiadať so situáciou *B*, za predpokladu, že merané hodnoty z akcelerometra boli dostatočne výrazné, čo bolo pri menších uhloch. Situácia *C* bola neprijateľná, kedže pri pohybe zariadenia nahor a nadol, akcelerometer v zložke z nezaznamenával dostatočne výrazné hodnoty.



Obr. 3.8: Rôzne typické naklonenia zariadenia.

V tejto práci ponúkame riešenie takejto situácie, ktoré je založené na spracovávaní údajov zo všetkých troch zložiek akcelerometra. Ak budeme merané dáta z akcelerometra reprezentovať ako vektory (vidieť na predošlých obrázkoch), môžeme si definovať vektor, ktorý bude súčtom vektorov všetkých troch zložiek (obrázok č. 3.9 - zložky, ktorých hodnoty sú blízke nule, nie sú zobrazené). Vektor súčtu (označený S) má smer vždy od zariadenia smerom nahor. Pre algoritmus detekcie krokov bude teda hodnotou z akcelerometra práve veľkosť tohto vektora.



Obr. 3.9: Vektor, ktorý je súčtom vektorov ostatných zložiek. Tento vektor smeruje vždy nahor a je preto nezávislý od naklonenia zariadenia.

Ak je zariadenie vo vodorovnej polohe A, tak tento vektor je takmer zhodný s vektorom zložky z. Aj keď by teoreticky mali byť tieto dva vektory totožné, v skutočnosti akcelerometer nie je úplne presný a produkuje aj šum. Teda pri pohybe nahor sa menia hodnoty všetkých troch zložiek aj keď niektoré minimálne. Z pohľadu algoritmov na detekciu krokov nejde o významný rozdiel.

Po implementácii a otestovaní sa ukázal tento prístup funkčný. Pri vodorovnom držaní zariadenia (situácia A) oba algoritmy detekovali rovnaké kroky ako v prípade, keď brali do úvahy len jednu zložku akcelerometra. Rovnako to bolo aj pri natočeniach B a C.

Hodnoty z kompasu sa trochu líšia pri rôznom naklonení zariadenia. Rozdiel medzi uhlom v situácii A a situácii C pri rovnakom smere chôdze bol v rozmedzí 15° až 30°. V porovnaní so samotnými nepresnosťami dát z kompasu sú tieto odchýlky akceptovateľné. Hodnoty zo senzora sú počítané pre situáciu A. Pre väčšinu používateľov je najtypickejšia pozícia B, kde je chyba v stupňoch zanedbateľná.

Kapitola 4

Lokalizácia s využitím akcelerometra a kompasu bez explicitne danej navigačnej cesty

Podmienka poznať navigačnú cestu je obmedzujúca. Ak sa používateľ rozhodne zmeniť smer trasy z akéhokoľvek dôvodu, musí byť znovu vypočítaná absolútna pozícia inou metódou. V tejto práci si kladieme otázku, či je možné lokalizovať používateľa aj bez explicitne danej navigačnej trasy. Znamenalo by to osamostatnenie algoritmu lokalizácie od navigácie. Hlavnou myšlienkou riešenia je určovanie pravdepodobnosti, že sa používateľ nachádza na nejakej pozícii v budove.

4.1 Bayesovské filtre

Indoor lokalizácia využívajúca detekciu krokov ponúka aj iné metódy ako je určovanie pozície vzhľadom k danej ceste. Na lokalizáciu sa často využívajú bayesovké filtre [24], ktoré predpokladajú Markovov model. V týchto riešeniach sa využíva pravdepodobnosť na odhad stavu nejakého dynamického systému. Existuje viacero implementácii bayesovských filtrov. Delia sa na spojité a diskrétne podľa toho ako reprezentujú systém. Najčastejšie používané sú nasledovné tri konkrétne implementácie, ktoré boli použité a porovnané aj vo výskume prezentovanom na konferencii IPIN [25].

- *Kalmanov filter* je najpoužívanejší z týchto filtrov (napr. [26]). Tento spojitý filter určuje pozíciu použitím normálneho rozdelenia. Stredná hodnota určuje aktuálnu pozíciu a rozptyl hovorí o presnosti odhadu. Jeho hlavnou výhodou je

výpočtová efektivita. Najlepšie výsledky dosahuje, ak ide o lineárny systém a ak neurčitosť pozície nie je veľmi vysoká.

- *Mriežkový prístup* je diskrétny filter, ktorý sa zakladá na rovnomernom rozdelení prostredia na mriežku s malými políčkami (typicky medzi 10cm a 1m), ktoré uchovávajú pravdepodobnosť, že sa používateľ nachádza na danej pozícii. Hlavnou výhodou je schopnosť reprezentácie akejkoľvek aj nelineárnej distribúcie pravdepodobnosti. Najväčšou nevýhodou je výpočtová zložitosť. Existujú prístupy, ktoré reprezentujú prostredie budovy pomocou grafov [27]. V takom prípade sa zvyšuje efektivita výpočtu, ale výsledok nemusí byť úplne presný.

- *Časticový filter* je diskrétna implementácia s využitím sekvenčnej Monte Carlo metódy (použitý napr. v tomto prístupe [8]). Reprezentuje pozície použitím množiny ohodnotených vzoriek (častíc). Tieto častice nasledujú pravidlá pohybového modelu. To znamená, že napríklad nemôžu prechádzať cez steny budovy v danom modeli. Najväčšou výhodou je schopnosť reprezentovať ľubovoľnú hustotu rozdelenia a je teda aplikovateľná aj na nelineárne systémy. V porovnaní s mriežkovým prístupom je efektívny v tom, že sa častice zameriavajú na oblasť s najväčšou pravdepodobnosťou. Problémom môže byť použitie vo viacrozmernom priestore, kedže zložitosť rastie exponenciálne s počtom dimenzií.

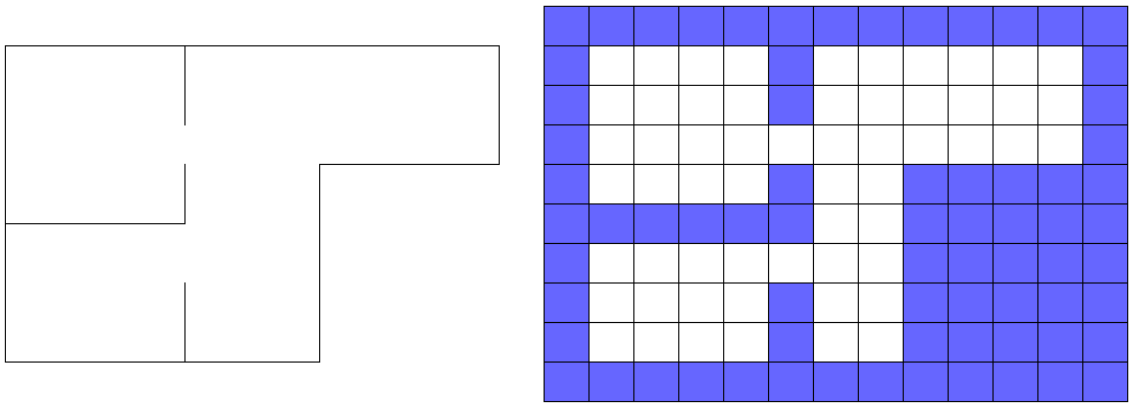
Určenie samotnej pozície pri použití kalmanovho alebo časticového filtra pozostáva z dvoch fáz. Prvou fázou je predikcia pozície na základe dostupných informácii o dĺžke kroku a natočení a druhou jej korekcia. Mriežkový prístup pracuje naopak. Najprv vypočíta pravdepodobnosť pre všetky miesta v budove a až potom sa určí aktuálna pozícia. Tým lepšie popisuje situáciu v budove, kedže si uchováva pravdepodobnosť pre všetky miesta. V porovnaní s predošlými filtrami sa teoreticky zdá byť odolnejši voči nepresnostiam z kompasu. Napríklad, keď sa objaví jedna alebo viacero hodnôt natočenia veľmi rozdielnych oproti skutočnému natočeniu. Táto situácia môže byť spôsobená prítomnosťou niečoho kovového. Obyčajná pracka opasku v kontakte so smartfónom môže spôsobiť odchýlku približne 50°. Tým, že mriežkový filter nie je upriamený svojou predikciou iba na tento konkrétny smer, tak by správnu informáciu o aktuálnej pozícii používateľa nemal stratiť. V konkrétnych riešeniach sa robí priemer z posledných niekoľkých hodnôt z kompasu, čo zamedzí tomu, aby niekoľko málo nepresných hodnôt z kompasu malo vplyv na presnosť lokalizácie.

Mriežkový prístup je používaný aj v oblasti robotiky [3]. Tiež ide o lokalizáciu, ale s niekoľkými rozdielmi v porovnaní s indoor lokalizáciou používateľa. Najpodstatnejší rozdiel je v tom, že pri indoor lokalizácii musíme počítať s rôznymi používateľmi, ktorí majú rôznu dĺžku kroku. Robot ma iné senzory ako smartfón. Ukázalo sa však, že tento prístup bol schopný sa vysporiadať s nepresnosťami zo týchto senzorov. Veľkou výhodou je aj schopnosť spracovať vstupy z viacerých senzorov.

Výzvou ostáva aplikovať mriežkový prístup na indoor lokalizáciu. Pri tom je potrebné myslieť na fakt, že ide o výpočtovo náročnú metódu. Cieľom je vytvoriť algoritmus lokalizácie a jeho implementáciu tak, aby bol schopný vypočítať aktuálnu pozíciu v reálnom čase. V praxi to znamená vypočítať pozíciu používateľa skôr ako vykoná ďalší krok, čo môže byť menej ako sekunda.

4.2 Lokalizačný pravdepodobnostný model

V tomto modeli reprezentujeme mapu budovy pomocou mriežky (obrázok 4.10).



Obr. 4.10: Reprezentácia mapy budovy pomocou mriežky. Modrou farbou sú vyznačené nedostupné políčka - steny a vpravo dole aj miesta mimo budovy.

Mriežka je dvojrozmerné pole, ktoré uchováva pravdepodobnosť, že sa používateľ nachádza na príslušnom mieste. Ak ide o viacposchodovú budovu, tak pre každé poschodie vytvoríme samostatnú mriežku. Predpokladáme, že vieme jednoznačne určiť poschodie, na ktorom sa používateľ nachádza. V tejto mriežke existujú dva typy políčok - dostupné a nedostupné. Dostupným políčkom nazývame každé také políčko, kde sa používateľ môže nachádzať na príslušnom mieste v budove. Nedostupným políčkom nazývame každé také políčko, kde sa používateľ nemôže nachádzať. Ide o steny budovy alebo nejaké prekážky v budove a miesta mimo budovy. Takáto definícia je vhodná

hlavne pri budovách, ktoré nemajú štvorcový alebo obdlžníkový pôdorys. Jedno poschodie vieme reprezentovať štvorcovou alebo obdlžníkovou maticou, kde miesta mimo budovy označíme ako nedostupné.

Dôležitou otázkou môže byť správna voľba veľkosti jedného políčka v mriežke. Určenie tejto hodnoty môže byť predmetom testovania.

Pre každé políčko mriežky sa uchováva hodnota pravdepodobnosti. $P[x][y]$ určuje pravdepodobnosť, že sa používateľ nachádza na príslušnej pozícii $[x, y]$ v budove na konkrétnom poschodí. $P[x][y] = 0$ znamená nulovú pravdepodobnosť, že sa používateľ nachádza na príslušnej pozícii. Takúto hodnotu nadobúdajú nedostupné políčka. Nulovú pravdepodobnosť môžu nadobúdať aj dostupné políčka, ak vieme s určitosťou povedať, že sa tam používateľ nenachádza.

Výpočet

Pred spustením výpočtu a detekovaním krokov predpokladáme mapu budovy reprezentovanú dvojrozmerným poľom. Táto mriežka má definované nedostupné políčka. Počiatkom výpočtu môžeme nazvať situáciu, keď určíme absolútnu pozíciu v budove niektorou metódou. Príslušná hodnota v poli P je nastavená na 1. Čím bude veľkosť políčka menšia, tým je väčšia pravdepodobnosť, že nebudeme vedieť stopercentne určiť aktuálnu pozíciu. V takom prípade môžeme viacerým susedným políčkam na danom mieste priradiť nenulovú pravdepodobnosť.

Samotný výpočet prebieha v iteráciach. Jedna iterácia nastane, keď algoritmus na detekciu krokov zaznamená krok. Rovnako v tom momente poskytne informáciu z kompasu o aktuálnom natočení. Algoritmus výpočtu dostane v každej iterácii hodnotu natočenia ako vstupný parameter. Po každej iterácii musí platiť, že súčet všetkých hodnôt pravdepodobnosti v mriežke je rovný 1. Pozícia, ktorá prislúcha políčku s najvyššou pravdepodobnosťou bude prehlásená za aktuálnu pozíciu používateľa.

Maska

Maska je dvojrozmerná štvorcová matica nepárnej dĺžky. Keď aplikujeme masku dĺžky $2r+1$ na políčko $P[x][y]$, znamená to, že $maska[r+1][r+1]$ (stredové políčko masky) hovorí aká je pravdepodobnosť toho, že ak bol používateľ na pozícii $[x, y]$, tak po detekovaní kroku je stále na pozícii $[x, y]$ a teda v skutočnosti neurobil krok. Pre niektoré iné políčko masky, napr. $maska[r + 1 + i][r + 1 + j]$ hovorí aká je pravdepodobnosť, že ak bol používateľ na aktuálnom políčku $[x, y]$, tak po detekovaní kroku sa dostal

na pozíciu $[x+i, y+j]$. Hovoríme, že ide o rozdávajúcu masku, pretože maska popisuje spôsob, akým rozdeľujeme hodnoty pravdepodobnosti medzi políčka v blízkosti aktuálnej polohy. Samotná maska nezávisí od políčka, na ktoré ju aplikujeme, ale od natočenia kroku a veľkosti jedného políčka mriežky. Znamená to, že rovnakú masku môžeme aplikovať na každé políčko mriežky.

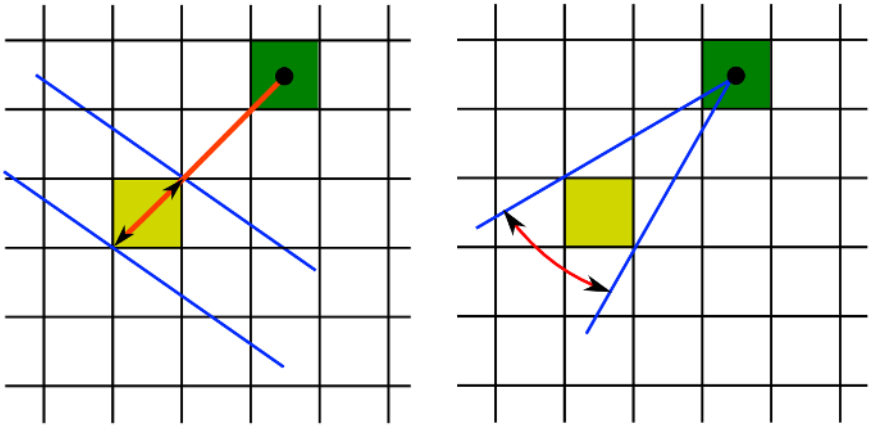
	0.02	0.07	0.1	0.07	0.02	
	0.03	0.1	0.2	0.1	0.03	
	0.01	0.05	0.1	0.05	0.01	
		0.01	0.02	0.01		

Obr. 4.11: Ukážka masky pre uhol 0°. Oranžovou farbou je zobrazené stredové políčko. Modrou farbou je zobrazené políčko s najväčšou pravdepodobnosťou. Prázdne políčka majú nulovú pravdepodobnosť.

Pre výpočet hodnoty jedného políčka masky potrebujeme určiť vzdialenosť od stredového políčka, čiže toho políčka, ktoré bude pri samotnom výpočte reprezentovať aktuálnu pozíciu a uhol, teda smer natočenia, v ktorom sa dané políčko nachádza vzhľadom na aktuálnu pozíciu. Tieto hodnoty sú nevyhnutné pre výpočet masky. Budeme predpokladať, že dĺžka kroku má normálne rozdelenie. Z distribučnej funkcie rozdelenia vieme určiť pravdepodobnosť, že dĺžka kroku bude z nejakého zadaného intervalu. Stredná hodnota dĺžky kroku bude nemenná. Rovnako budeme predpokladať, že uhol natočenia má normálne rozdelenie. Strednou hodnotou bude v danom prípade aktuálne nameraná hodnota z kompasu.

Pre jednotlivé políčka masky počítame pravdepodobnosť, že sa ak bol používateľ na stredovom políčku, tak po detekovaní kroku sa dostal na dané políčko. Pri tom využijeme hodnoty z distribučných funkcií a to tak, že budeme zisťovať pravdepodobnosť, že dĺžka kroku je z intervalu, ktorý zodpovedá minimálnej a maximálnej vzdialenosti medzi danými dvoma políčkami (obrázok č. 4.12). Dolná hranica intervalu bude vzdialenosť od stredu aktuálneho stredového políčka k najbližšiemu bodu z počítaného políčka. Horná hranica bude vzdialenosť od stredu políčka k najvzdialenejšiemu bodu v počítanom políčku. Rovnako to platí pre natočenie, kde sa berie do

úvahy najmenší a najväčší uhol, ktorý smeruje zo stredu aktuálneho stredového políčka a prechádza aj aktuálnym počítaným políčkom. Veľkosť masky závisí od veľkosti políčka mriežky a maximálneho možného kroku.

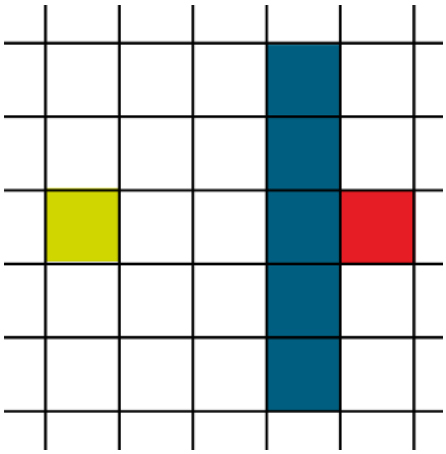


Obr. 4.12: Výpočet pravdepodobnosti pre žlté políčko masky vzhľadom na zelené políčko. Na prvom obrázku je znázornený interval dĺžky kroku a na druhom interval natočenia, teda uhla získaného z kompasu, ktorý znamená, že sa používateľ dostane zo zeleného políčka na žlté.

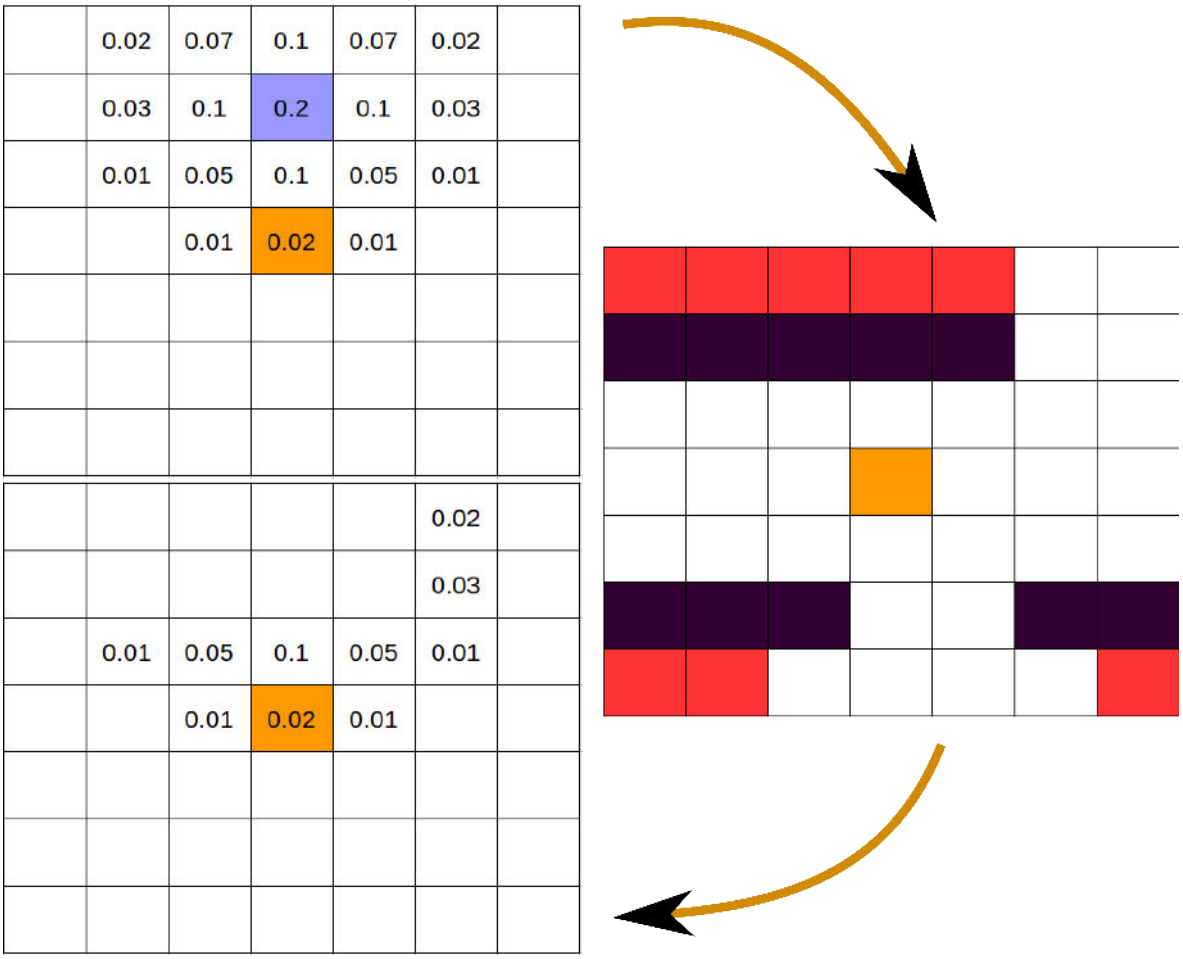
Povrchová maska

Nami definovaná univerzálna maska (nezávislá od mriežky) môže byť aplikovaná na akékoľvek políčko mriežky za predpokladu, že je možné dostať sa zo stredového políčka na ktorékoľvek iné políčko patriace maske. Problém nastáva pri prítomnosti nedostupných políčok, čiže napríklad v blízkosti stien. Samotné nedostupné políčka nie sú problémom a dá sa to vyriešiť jednoduchou podmienkou. Zaujímava situácia nastáva, ak je nejaké políčko dostupné, ale medzi ním a stredovým políčkom sa nachádza nedostupné miesto. Univerzálna neupravená maska by priradila pravdepodobnosť aj tomuto políčku, čo by znamenalo, že používateľ môže prechádzať cez steny. Situáciu vidíme na obrázku 4.13.

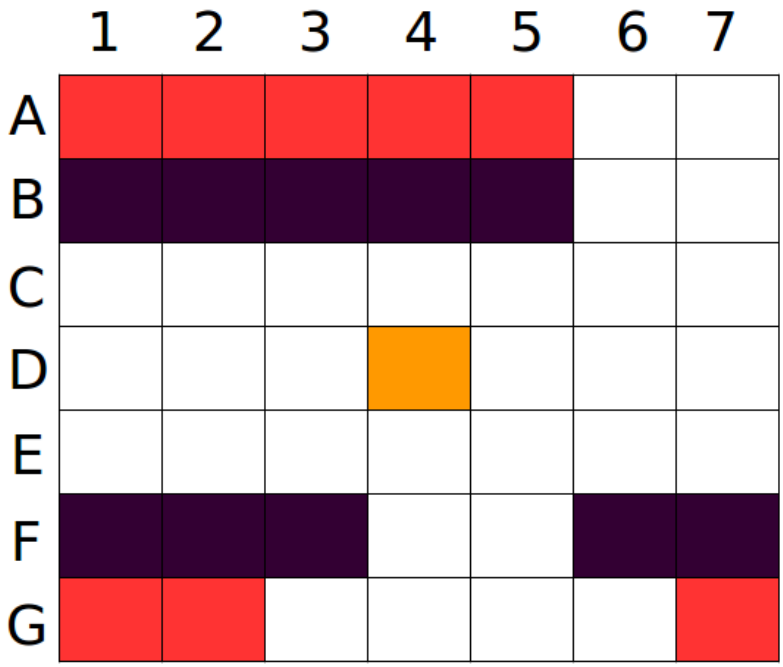
V určitých prípadoch je potrebné masku upraviť podľa konkrétneho miesta, kde sa aplikuje. Pre potreby nášho algoritmu sme si takúto upravenú masku nazvali povrchová maska. Ide teda o masku rovnakých rozmerov, ktorá má oproti pôvodnej maske niektoré hodnoty rovné nule. Povrchom nazveme dvojrozmerné pole takej istej veľkosti, ktoré pre každé políčko určuje, či je dostupné alebo nedostupné vzhľadom na mriežku aj na stredové políčko. Tieto tri polia sú zobrazené na obrázku 4.14.



Obr. 4.13: Časť masky s problémom prechádzania cez steny. Modrou farbou je znázornená stena, čiže nedostupné políčka. Žltá farba reprezentuje stredové políčko. Červenou farbou je znázornené jedno z políčok, ktoré by v danej situácii malo mať nulovú pravdepodobnosť aj napriek tomu, že je dostupné.



Obr. 4.14: Vľavo hore je maska. Vpravo je povrch. Vľavo dole je povrchová maska.



Obr. 4.15: Ukážka povrchu. Tmavou farbou sú zobrazené nedostupné políčka. Žltá farba reprezentuje stredové políčko. Červenou farbou sú znázornené políčka, ktoré označíme ako nedstupné vzhľadom na aktuálne stredové políčko.

Na obrázku 4.15 vidíme príklad povrchu. Pre každé políčko sa určí, či je dostupné vzhľadom na stredové políčko D4. Pre niektoré typické situácie, popíšeme ako sa určuje, či je dané políčko dostupné.

- D2 - je dostupné, lebo medzi stredovým a týmto políčkom nie je žiadna prekážka.
- B4 - je nedostupné. Vyplýva to zo samotnej mriežky, ktorá je reprezentáciou budovy. Na tomto mieste môže byť napríklad stena.
- A4 - je nedostupné. Ak sledujeme stĺpec číslo 4, tak vidíme, že A4 je nedostupné z políčka D4 práve kvôli inému nedostupnému políčku B4.
- G1 - je nedostupné, pretože ak sledujeme diagonálu, tak medzi G1 a D4 sa nachádza políčko F2, ktoré je nedostupné.
- A3 - je nedostupné. Nevieme sledovať presne nejaký stĺpec, riadok alebo diagonálu. Preto berieme do úvahy dve políčka B3 a B4. Obidve tieto miesta sú nedostupné, tak aj políčko A3 označíme za nedostupné. Rovnako to je aj napr. s políčkom G2. V tomto prípade posudzujeme F2 a F3.

- G6 - je dostupné. Rovnakým spôsobom ako pre predošlé prípady sa pozrieme na dve susedné políčka smerom k stredu. Konkrétne F5 a F6. Je postačujúce ak je jedno z týchto políčok dostupné. V našom prípade ide o F5 a teda G6 označíme ako dostupné. Rovnako je to s políčkom A6 (pozeráme sa na B5 a B6) alebo G3 (F3, F4). Ak sú obidve posudzované políčka dostupné, tak aj aktuálne políčko prehlásime za dostupné ako v prípade B7 (posudzované políčka B6 a C6).

Poznamenajme, že pri posudzovaní dostupnosti sa berú do úvahy nedostupné políčka, ktoré sú takými z charakteristiky budovy, ale aj tie nedostupné políčka, ktoré boli určené v tomto procese vytvárania povrchu vzhľadom na stredové políčko. Je dôležité rozhodovať o dostupnosti políčka najprv pre miesta bližšie k stredu, čiže jeho susedné políčka. Potom ich susedné políčka a nakoniec aj tie v ďalších úrovniach.

Algoritmus výpočtu

Počas jednej iterácie je potrebné aktualizovať pole uchovávajúce hodnoty pravdepodobností. Algoritmus pracuje s dvoma stavmi poľa pravdepodobností. Pri každej iterácii máme k dispozícii pole P, ktoré je nemenné. Všetky výpočty používaju toto pole, ale výsledky zapisujú do poľa P'. Pole P' sa pri ďalšej iterácii stáva nemenným poľom P. Vidieť to v nasledovnom algoritme 1.

Na začiatku po detekovaní kroku by sa mala vypočítať maska, v skutočnosti sa však iba načíta vopred vytvorená maska pre daný uhol. Kedže hodnoty z kompasu nie sú úplne presné, tak si môžeme uchovávať konečný počet masiek a získaný uhol zaokrúhlime k danej hodnote. Masku aplikujeme na nejake políčko iba v prípade, že je dostupné a hodnota pravdepodobnosti je nenulová. Kedže ide o rozdávajúcu masku, nulová pravdepodobnosť by znamenala, že sa hodnoty susedných políčok nebudú meniť. Následne sa vytvára povrchová maska na základe povrchu pre dané políčko, čo je tiež vypočítané na začiatku a pri každej iterácii sa toto pole iba načíta. Na konci celého algoritmu je potrebné normalizovať celú mriežku, aby súčet pravdepodobností bol rovný 1. Samotnú masku nenormalizujeme. Políčko s najväčšou hodnotou pravdepodobnosti prislúcha aktuálnej pozícii používateľa.

```
1  Input: P - pole pravdepodobností veľkosti X × Y
2         s - uhol natočenia získaný z kompasu
3  Output: P' - pole pravdepodobností veľkosti X × Y
4  maska ← vypočítajMasku(s);
5  r ← |maska|/2;
6  inicializuj P'[X][Y];
7  for x ← 0 to X do
8      for y ← 0 to Y do
9          if P[x][y] je dostupné and P[x][y] > 0 then
10             povrch ← získajPovrch(x, y);
11             povrchovaMaska ← upravPodľaPovrchu(povrch, maska);
12             for i ← (x − r) to (x + r) do
13                 for j ← (y − r) to (y + r) do
14                     P'[i][j] ← P'[i][j]+P[x][y]·povrchovaMaska[i−x+r][j−y+r]
15                 end
16             end
17         else
18             P'[x][y] ← 0;
19         end
20     end
21 end
22 P' ← normalizuj(P');
23 return P'
```

Algoritmus 1: Aplikovanie masky. Výpočet prislúchajúci jednej iterácii, teda jednému detekovanému kroku.

Kapitola 5

Efektívna implementácia

Pri implementovaní algoritmov na smartfónoch treba mať na zreteli ich obmedzený výpočtový výkon. Podstatné je, aby výpočet aktuálnej pozície prebiehal v reálnom čase. Navrhnutý lokalizačný model môžeme zefektívniť a zlepšiť jeho časové aj pamäťové nároky.

5.1 Implementačné detaily

V lokalizačnom pravdepodobnostnom modeli bola pozícia definovaná dvoma súradnicami v mriežke, ktorá reprezentuje mapu budovy. Je potrebné spomenúť viacposchodové budovy. Mapou v našom modeli budeme rozumieť mapu jedného poschodia budovy. Nie je potrebné zväčšovať pole pravdepodobností o ďalší rozmer. Prechod medzi poschodiami je možný len na niektorých špeciálnych miestach v budove, konkrétne na schodoch alebo výťahoch. Pri zmene poschodia môžeme aplikovať algoritmus určenia absolútnej pozície v budove.

Zaujímavou otázkou je určenie veľkosti jedného políčka mriežky. Čím je políčko menšie, tým je lokalizácia presnejšia. Spolu s presnosťou rastie aj náročnosť výpočtu, kedže mriežka obsahuje viac políčok. Ak je políčko príliš veľké, môže dôjsť k výraznému poklesu kvality lokalizácie. Napríklad, ak máme políčko dĺžky jeden meter, tak pri detekovaní kroku môžu byť kroky dĺžky až do 70 centimetrov interpretované, akoby používateľ ostal na danom mieste (dĺžka od stredu políčka k najvzdialenejšiemu bodu políčka je približne 70 centimetrov). V spomínanej práci z oblasti robotiky bolo zvolené štvorcové políčko so stranou dlhou 15 centimetrov. V našom prípade sme zvolili veľkosť 30 centimetrov. Pri tejto hodnote krátke kroky nebudú ignorované. Charakterizovanie pozície používateľa pomocou takto veľkého políčka vyzerá byť prirodzené,

kedže veľkosť chodidla priemerného človeka je len o pár centimetrov menšia.

Pri výpočte masky je potrebné určiť strednú hodnotu a rozptyl pre normálne rozdelenie dĺžky kroku a uhla natočenia. Tieto hodnoty nemusia byť určené úplne presne. Algoritmus by mal byť schopný sa vysporiadať s rôznou dĺžkou kroku. Strednú hodnotu dĺžky kroku sme zvolili $70cm$. Rozptyl bol zvolený tak, aby bola $99,7\%$ pravdepodobnosť, že dĺžka kroku je z intervalu medzi 45 a 115 centimetrov. Pri normálnom rozdelení, ktoré zodpovedá uhlu natočenia, je strednou hodnotou tá hodnota, ktorá je získaná z kompasu pri detekovaní kroku. Rozptyl je zvolený tak, aby bola 99.7% pravdepodobnosť, že uhol sa líši o menej ako 45° v oboch smeroch. To by malo zaručiť správne fungovanie algoritmu pri priamočiarej chôdzi bez zmeny smeru alebo pri konštantne nepresných hodnotách spôsobených napríklad iným naklonením zariadenia.

Maska sa negeneruje pri detekovaní kroku. Kedže maska je nezávislá od políčka, na ktoré bude aplikovaná, tak masky budú pre všetky uhly vygenerované vopred. Uchováva sa maska pre každý uhol s presnosťou na jeden stupeň, čiže celkovo 360 masiek. Táto hodnota by mohla byť aj menšia. Pri týchto hodnotách je postačujúce mať štvorcovú masku s dĺžkou 11 políčok. Teda v jednej maske je 121 políčok.

5.2 Lokalizačný pravdepodobnostný model z pohľadu implementácie

Aktuálne navrhnutý model predpokladá dvojrozmerné pole s hodnotami pravdepodobností zodpovedajúce veľkosti mapy. Potrebujeme uvažovať aj druhé rovnako veľké pole, ktoré je potrebné pri výpočte. Samotný výpočet prebieha v iteráciach, kde sa z jedného poľa hodnoty čítajú a do druhého zapisujú (polia P a P'). Veľkosť týchto polí závisí od veľkosti políčka mriežky. Napríklad, ak máme zvolenú veľkosť políčka 30 centimetrov a máme budovu so štvorcovým pôdorysom, kde dĺžka jedného rozmeru je 30 metrov ($900m^2$), tak jedno pole s hodnotami pravdepodobnosti obsahuje 10000 políčok. Pre každé políčko prislúchajúce pozícii na mape je potrebné si pamätať tieto tri hodnoty:

1. dostupnosť políčka (hodnota *true* alebo *false*)
2. hodnotu pravdepodobnosti
3. povrch (dvojrozmerné pole rovnakej veľkosti ako maska) - túto informáciu nemusíme mať uloženú, avšak povrch sa počas behu výpočtu nemení.

Prvé dve informácie vieme zlúčiť do jednej. Hodnota pravdepodobnosti je z intervalu $<0, 1>$. Nedostupné políčka môžeme odlíšiť od dostupných tak, že im priradíme zápornú hodnotu pravdepodobnosti, napríklad -1.

Mapu nebudeme reprezentovať pomocou dvojrozmerného poľa, ale ako jednorozmerný zoznam políčok. Ak máme mriežku rozmerov W a H, tak budeme mať jednorozmerné pole dĺžky $W \times H$. Pozícia $[i, j]$ v dvojrozmernej reprezentácii prislúcha indexu $[i \cdot W + j]$ v jednorozmernom poli.

Maska

Hodnoty v maske (na rozdiel od povrchovej masky) nezávisia od miesta, kde ju aplikujeme. Jediná premenná, ktorá ovplyvňuje hodnoty v maske, je veľkosť uhla natočenia získaného z kompasu. Nie je potrebné počítať všetkých 360 masiek. Postačí štvrtina a pre ďalšie uhly iba otočíme prislúchajúcu masku.

Pri rozmere masky 11 políčok je spolu v maske 121 hodnôt. Po výpočte masiek pre všetky definované uhly si môžeme všimnúť, že približne polovica hodnôt v maske je nulová (53 až 59 hodnôt). Zo zvyšných hodnôt je iba 14 až 17 hodnôt väčších ako 0.001. Zvyšných 46 až 54 hodnôt je teda nenulových, ale veľmi malých. Zefektívnenie masky môže znamenať, že masku nebudeme reprezentovať ako dvojrozmerné pole s polovicou nulových hodnôt, ale ako jednorozmerný zoznam nenulových hodnôt. Testovaním sa ukázalo, že nie je možné vynechať malé hodnoty, ktoré zdanlivo nemajú vplyv na celkovú distribúciu pravdepodobnosti. Problém sa ukázal v blízkosti stien, kde sa za určitých okolností objavili v mriežke iba nulové hodnoty a stratila sa informácia o aktuálnej pozícii, čo bolo spôsobené práve tým, že malé hodnoty z masky boli zamenené za nulové hodnoty.

Pri tejto možnosti sa nebude počítať povrch a povrchová maska, ale pre každé políčko masky sa uloží zoznam indexov, na ktoré sa treba pozrieť pri overovaní dostupnosti políčka.

Maska bude teda reprezentovaná zoznamom políčok, kde pre každé políčko je potrebné pamätať si tieto tri veci:

1. Index tohto políčka - tu neide o index v rámci masky. Táto hodnota je skôr indexom posunu oproti stredovému políčku vzhľadom na mapu.

2. Hodnotu pravdepodobnosti - vieme o nej povedať, že je nenulová.

3. Zoznam indexov políčok, na ktoré sa pri výpočte dostupnosti treba pozrieť - tiež ide o posun oproti stredovému políčku vzhľadom na mapu.

V tomto bode už maska prestáva byť úplne nezávislá od mapy, na ktorú ju aplikujeme. Pri výpočte masky je potrebné poznať rozmer mriežkovej reprezentácie mapy, aby sme vedeli vypočítať indexy v maske. Maska je naďalej nezávislá od konkrétneho políčka mapy, na ktoré je aplikovaná. Pri každom načítaní mapy je teda možné vypočítať a uložiť si masky pre všetky uhly, ktoré uvažujeme.

Výpočet

Pri aplikovaní masky na dané políčko mriežky je potrebné overiť, či je dané políčko dostupné. Pre zjednodušenie tohto výpočtu sme upravili algoritmus overenia dostupnosti políčka. Majme masku so stredovým políčkom na súradniciach $[x_s, y_s]$ a políčko, pre ktoré overujeme dostupnosť na pozícii $[x_p, y_p]$. Nech platí, že $x_s \leq x_p$ a $y_s \leq y_p$. Potom algortimus bude hľadať políčka, ktoré sú na ceste medzi stredovým a cieľovým políčkom. Pri dvojrozmernej maske by algoritmus vyzeral nasledovne 2.

```
1  Input: M - maska rozmerov l × l
2         x_p, y_p - súradnice hľadaného políčka
3  Output: C - množina políčok na ceste medzi stredovým a cieľovým políčkom
4  inicializuj C;
5  x ← l/2;
6  y ← l/2;
7  while x < x_p and y < y_p do
8      if x < x_p then
9          x++;
10     end
11     if y < y_p then
12         y++;
13     end
14     C ← C ∪ {[x, y]}
15 end
16 return C
```

Algoritmus 2: Hľadanie políčok, ktoré podmieňujú dostupnosť cieľového políčka.

Ak ide o políčko, ktoré má jednu alebo obidve súradnice menšie ako stredové políčko, tak sa bude hodnota x alebo y odpočítavať. Tento algoritmus implementujeme s jednorozmernou úplnou maskou, čiže ešte pred odstránením nulových hodnôt. Pri samotnom výpočte je potom potrebné overiť, či je políčko dostupné a v tom prípade mu prideliť príslušnú pravdepodobnosť. Tento algoritmus môže inak určiť dostupné políčka ako predošlý algoritmus. Avšak ide iba o akési okrajové políčka zväčša pri rohoch stien. Ak je políčko za stenou, tak oba algoritmy ho správne označia ako nedostupné.

Po obvode mriežky pridáme niekoľko nedostupných políčok, aby nebolo nutné počas aplikovania masky stále overovať, či dané políčko vôbec existuje.

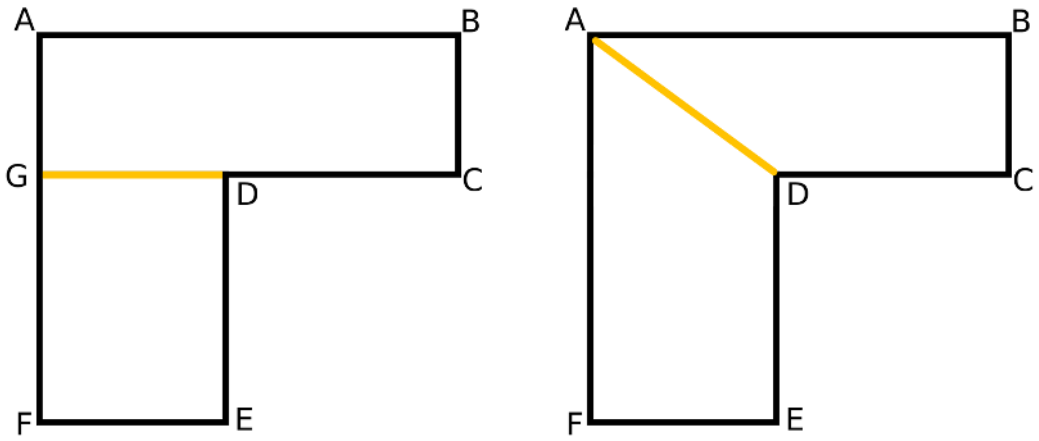
Ďalšou zmenou, ktorá môže priniesť zefektívnenie, je pamätanie si indexov políčok v mriežke s nenulovou hodnotou pravdepodobnosti. Nie je teda potrebné prechádzať všetkými políčkami mriežky a aplikovať na ne masku. Do úvahy sa zoberie iba zoznam nenulových políčok. Okrem samotného výpočtu pravdepodobností sa zrýchlia aj ostatné pomocné výpočty, ako je normalizácia mriežky alebo určenie aktuálnej pozície, čiže políčka s najväčšou pravdepodobnosťou. Tento prístup poskytuje možnosť ignorovať políčka s malými hodnotami alebo vybrať iba určitý počet políčok s najväčšou pravdepodobnosťou.

5.3 Generovanie a uchovávanie máp

Pri návrhu riešenia pre indoor lokalizáciu sme kládli dôraz aj na to, aby bolo možné doplniť ďalšie prvky indoor navigačnej aplikácie. Kľúčovou otázkou pri implementácii je spôsob vytvárania a uchovávania máp. Mapu používateľovi budeme zobrazovať vo formáte *SVG*, ktorý má svoje výhody hlavne z pohľadu pamäte. Aktuálna pozícia bude vykreslená v tomto zobrazení mapy. Je tu priestor pre zobrazenie ďalšej vrstvy informácii na túto mapu, napríklad zobrazenie navigačnej cesty. Pre lokalizačný pravdepodobnostný model je nutné vytvoriť mriežku, ktorá zodpovedá mape budovy. Táto mriežka by mala byť generovaná automaticky z popisu mapy.

Mapa bude definovaná zónami, čo sú v podstate body a spojnice medzi nimi (obrázok 5.16). V tomto prístupe vystupuje niekoľko prvkov:

- *Body* sú základnou jednotkou mapy. Nesú informáciu o pozícii v budove, čiže sú definované súradnicami v metroch alebo centimetroch. Uchovávajú sa iba význačné body v budove. Ich počet závisí od štruktúry budovy.



Obr. 5.16: Ukážka dvoch rôznych reprezentácii rovnakej mapy pomocou zón.

- *Spojnice* spájajú body. Väčšinou ide o steny. Môžu však existovať aj priechodné spojnice, napríklad dvere do miestnosti alebo priehľadné spojnice pre potreby uzavretia zóny, ako vidieť na obrázku, kde sú zobrazené oranžovou farbou. Spojnica je definovaná dvojicou bodov, ktoré spája. Konkrétne môže ísť o dva indexy v zozname bodov. Ďalšou informáciou je typ spojnice.

- *Zóny* sú podstatnou súčasťou mapy. Zóna vznikne spojením viacerých spojníc. Môže byť definovaná ako zoznam bodov alebo zoznam spojníc. Pre potreby navigácie predpokladáme, že zóny sú konvexné mnohouholníky. Ak nejaká zóna nie je konvexná, tak ju vieme rozdeliť na dve alebo viacero konvexných zón. Na obrázku vidíme nekonvexnú zónu definovanú čiernymi spojnicami. Vieme ju rozdeliť na dve konvexné zóny pridaním priehľadnej spojnice zobrazenej oranžovou farbou. Existuje viacero správnych spôsobov. Jeden z nich si vyžadoval pridanie dodatočného bodu (bod G na obrázku vľavo). Zóna môže tiež obsahovať informáciu o svojom type. Niektoré zóny sú špecifické, napríklad ak ide o schodisko.

- *Miestnosti* sú akýmisi logickými jednotkami na mape budovy. Ide o zlúčenie viacerých zón. Miestnosť môže napríklad vzniknúť spojením zón, ktoré tvoria nejakú chodbu. Miestnosti môžu uchovávať zóny, ktoré ich tvoria alebo každej zóne pridáme informáciu, do ktorej miestnosti patrí. V prípade našej indoor lokalizácie tieto miestnosti nevyužívame. Môže to byť ale zaujímavá informácia pre potreby zobrazovania alebo samotnej navigácie.

Každá mapa môže byť uchovaná v týchto troch súboroch:

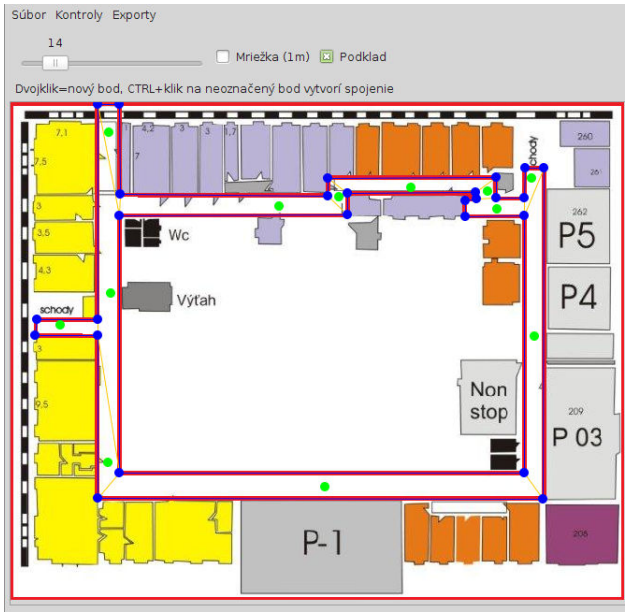
- *Základný popisný súbor* môže byť v textovom alebo xml formáte. Uchováva celkové rozmery mapy v metroch alebo centimetroch. Ak je potrebné, tak môžeme pridať aj uhol otočenia celej mapy, ak tá je natočená inak ako v skutočnosti. Tu môžu byť vložené aj ďalšie informácie o mape.

- *SVG súbor* je určený na zobrazenie používateľovi. Tento súbor by mal byť pre používateľa esteticky príjemný a čitateľný. Môžu byť použité rôzne farby, značky alebo piktogramy na identifikáciu nejakých zaujímavých miest v budove.

- *Binárny súbor* uchováva informácie o mape, ktoré sú potrebné pre generovanie mriežky. Binárny formát bol zvolený pre rýchlosť načítania všetkých potrebných informácii a aby tento súbor mal čo najmenšiu veľkosť. Tieto informácie nemusia byť čitateľné pre používateľa. V tomto súbore uchovávame iba zoznam bodov s ich súradnicami, zoznam zón, ktoré obsahujú indexy príslušných bodov a ďalšie informácie o type zóny a typoch jednotlivých spojníc.

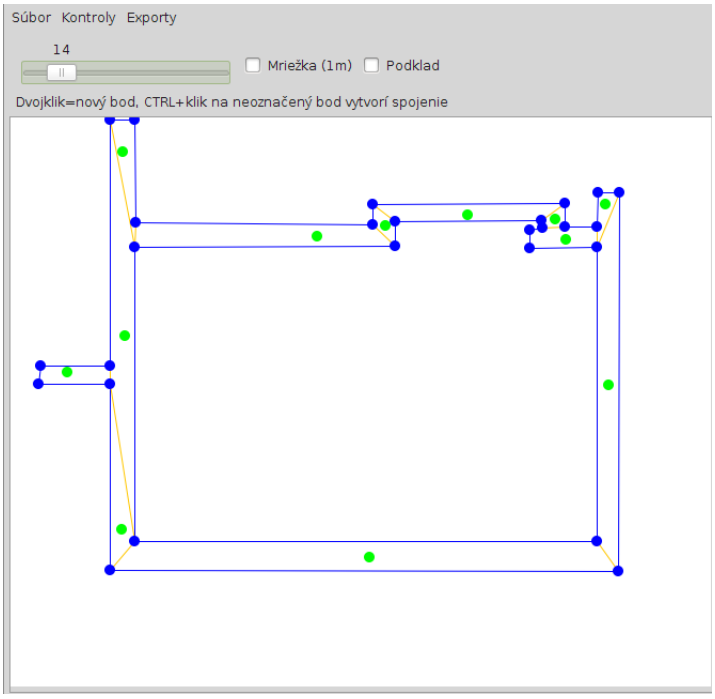
Vytváranie máp

Vytvorili sme editor máp, ktorý poskytuje možnosť vytvorenia informácii o mape. Body, spojnice ani zóny nie sú detekované automaticky. Používateľ si môže načítať



Obr. 5.17: Editor máp, ktorý poskytuje možnosť vytvorenia informácii o mape na základe podkladu.

podkladový *SVG* súbor a jednoducho zadať body a spojnice medzi nimi s informáciou o ich type. Rovnako aj zóny a všetky dodatočné charakteristiky. Pozícia bodov je počítaná automaticky. Editor máp poskytuje možnosť automatického overenia konvexnosti zón. Pri tomto overovaní bol použitý fakt, že konvexný mnohouholník má všetky uhly menšie ako 180°. Tento editor robí aj export zadaných informácii do binárneho súboru.

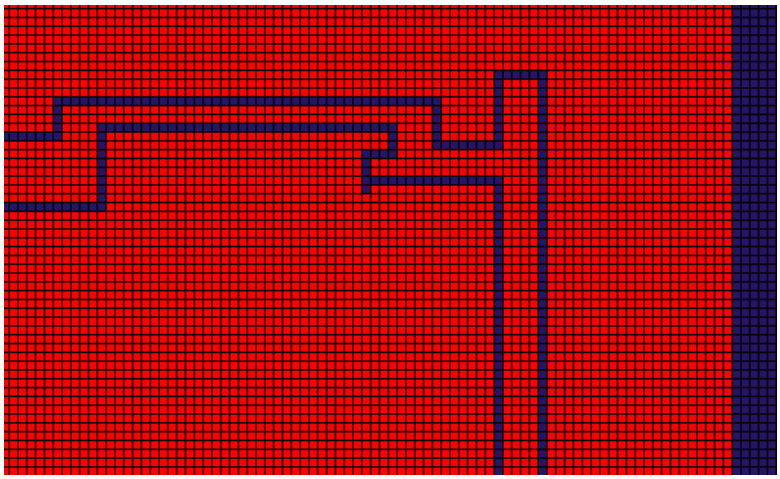


Obr. 5.18: Zobrazenie informácii o mape bez podkladového súboru. Zelenou sú zobrazené body, ktoré uchovávajú informáciu o zóne, do ktorej patria.

Generovanie mriežky

Pri generovaní mriežky je potrebné poznať rozmery celej mapy, ktoré sa získajú zo základného popisného súboru. Mriežka je potom generovaná na základe informácii z binárneho súboru. Do mriežky sa postupne zaznamenávajú zóny. Cieľom je získať pole pravdepodobností, kde budú označené dostupné políčka hodnotou 0, čiže políčka vo vnútri zón. Zvyšné políčka budú označené ako nedostupné, čo znamená, že im bude priradená záporná hodnota -1. Na začiatku obsahuje mriežka políčka s hodnotou -2, okrem políčok po obvode, ktoré boli pridané v rámci efektívnej implementácie. Tie majú hodnotu -1. Pre každú zónu sa vypočíta index každého bodu, ktorý definuje danú zónu. Použitím jednoduchého známeho algoritmu na kreslenie čiar sa vyplnia

nedostupné políčka prislúchajúce nepriechodnej spojnici. Po vykreslení všetkých stien sa detekuje bod vo vnútri každej zóny. Ten sa vypočíta ako priemer prvého a tretieho bodu v zóne. Podmienka konvexnosti zóny zaručuje, že takto vypočítaný bod bude vo vnútri zóny. Použitím algoritmu semienkového vypĺňania sa nastavia hodnoty pravdepodobností vo vnútri zóny na hodnotu 0. Ak zóna obsahuje priechodnú stenu, tak sa zaplní aj susedná zóna. Na konci sa všetky nenulové políčka prehlásia za nedostupné.



Obr. 5.19: Časť mapy reprezentovaná mriežkou.

5.4 Časová a pamäťová zložitosť

Pre prvotné overenie algoritmu sme implementovali lokalizačný pravdepodobnostný model tak, ako bol definovaný. Výsledkom nebol prototyp pre smartfón, ale aplikácia určená pre počítač. Na nej sme pomocou simulovaných dát zo senzorov sledovali správanie celého modelu. Následne bola iplementovaná efektívnejšia verzia opísaná v tejto kapitole. Tieto algoritmy sme implementovali aj vo forme prototypu na platformu Android.

Čas výpočtu

Časová zložitosť algoritmu výpočtu pravdepodobností po detekovaní kroku je lineárna vzhľadom na počet políčok v mriežke. Veľkosť masky je zanedbateľná v porovnaní s veľkosťou mriežky. Rýchlosť výpočtu sme overili aj pre konkrétnu mapu budovy. Na testovanie bol použitý notebook Asus UL 30 JT (Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz, DDR3 1333 MHz SDRAM, Linux Mint 16). Nami navrhnutú efektívnu

implementáciu sme testovali na smartfóne HTC Wildfire S (600 MHz ARM 11, 512 MB RAM, Android OS v2.3).

Výpočet poľa pravdepodobností po detekovaní kroku pozostáva z dvoch krokov. Prvým je aplikovanie masky na políčka s nenulovou hodnotou. Druhým krokom je normalizácia mriežky. Prvý krok závisí od počtu nenulových políčok, čo je ovplyvnené štruktúrou budovy. Pri normalizovaní mriežky v prvej implementácii je potrebné prejsť všetky políčka v mriežke. V druhej implementácii si uchovávame zoznam indexov s nenulovou hodnotou. Pri počte políčok 20000, ktoré majú nenulovú hodnotu, prvý algoritmus dokázal vykonať výpočet v čase približne $50ms$ a druhý v čase približne 40ms. Druhá implementácia priniesla len minimálne zlepšenie. Mapa mala rozmery približne 50 a 40 metrov. Pri väčšej mape by tieto rozdiely mohli byť výraznejšie. Ukázalo sa však, že výpočet je možné robiť v reálnom čase. Podobné výsledky dosahoval aj prototyp na smartfóne.

Ďalšou významnou operáciou je výpočet masiek hneď po načítaní mapy. Na notebooku generovanie masiek pre 360 uhlov trvalo približne $300ms$. Na smartfóne to trvalo viac ako 25 sekúnd. Po redukcii výpočtu na štvrtinu bol čas výpočtu približne $6800ms$. Zvyšné masky boli vyrátané z predošlých masiek ich otočením. Na výkonnejších smartfónoch alebo tabletoch neide o tak výrazný čas. Tento výpočet vieme odstrániť, kedže masky nie sú závislé od mapy. Bolo by potrebné uložiť si tieto masky a po vygenerovaní mriežky ich aj načítať.

Pamäťové nároky

Rozdiel medzi prvou a druhou implementáciou je výraznejší z pohľadu pamäťových nárokov ako z pohľadu času výpočtu. Budeme analyzovať iba objekty, ktoré je potrebné si uchovávať po celú dobu výpočtu. Na uloženie hodnôt pravdepodobností sme použili 4 bity (float), nie 8 bitov (double).

Mriežka reprezentuje mapu budovy. Nech n je počet políčok v mriežke. V prvej implementácii si uchováva dve dvojrozmerné polia so súčtom políčok n, kde sa nachádzajú aktuálne, resp. počítané hodnoty pravdepodobností. Ďalšie dvojrozmerné pole s počtom políčok n uchováva informáciu o dostupnosti. Pre každé políčko je tiež potrebné si pamätať povrch, čo je spolu $m \times n$ hodnôt, kde m je počet políčok v maske. V našom prípade je $m = 121$. Mriežka si spolu uchováva $2 \cdot 4n + 2n + 121 \cdot 2n = 252n$ bitov. Informácie o veľkosti mriežky alebo masky neuvažujeme. Ak máme mapu rozmerov 45×30 metrov, tak počet políčok v mriežke je 15000. Mriežka teda uchováva

približne $470kB$.

V druhej implementácii je tiež potrebné mať dve polia s hodnotami pravdepodobností. Tie sú však jednorozmerné. Uchováva sa aj zoznam nenulových políčok, ktorý závisí od štruktúry budovy. Tieto zoznamy sú dva, jeden hovorí o aktuálnom stave a do druhého sa zapisujú indexy počas výpočtu. Je potrebné poznamenať, že počet políčok mriežky je v tejto implementácii väčší kvôli pridaniu nedostupných políčok po obvode mapy. Konkrétne ide o $2b(2b + x + y)$ políčok, kde b musí byť viac ako polovica rozmeru masky (v našom prípade 6), x a y hovoria o rozmere mapy. Označme tento počet n'. Mriežka v tejto implementácii si uchováva $2 \cdot 4n' + 4p$ bitov, kde p je maximálny počet dostupných políčok. Táto hodnota je vypočítaná pri generovaní mriežky. V najhoršom prípade je $p = n$. Mriežka si teda uchováva $8n' + 4n$ bitov. Pre mapu rozmerov 45×30 metrov ide o približne $25kB$.

Druhou podstatnou časťou je uchovávanie masiek pre potreby výpočtu. V prvej implementácii je jedna maska definovaná dvojrozmerným poľom s počtom políčok 121 pre veľkosť políčka mriežky $30cm$. Spolu je potrebné si uchovávať $360 \cdot 121 \cdot 4$ bitov, teda takmer $22kB$. Pri druhej implementácii je maska definovaná hodnotou indexu posunutia oproti stredovému políčku, hodnotou pravdepodobností a zoznamom políčok, pre ktoré potrebuje overiť dostupnosť. Týchto políčok je najviac 6 vrátane aktuálneho políčka. Počet nenulových políčok masky je v priemere 50. Spolu je potrebné uchovávať $360 \cdot 50 \cdot (4 + 4 + 6 * 4)$ bitov, teda nie viac ako $72kB$.

45×30 metrov	Prvá implementácia	Druhá implementácia
Mriežka	$470kB$	$25kB$
Masky	$22kB$	$72kB$
Spolu	$492kB$	$97kB$

Veľkosť pamäte potrebná pre masky je nezávislá od veľkosti mapy. Majme teraz mapu rozmerov 90×60 metrov, teda štvornásobne viac políčok.

90×60 metrov	Prvá implementácia	Druhá implementácia
Mriežka	$1890kB$	$96kB$
Masky	$22kB$	$72kB$
Spolu	$1,9MB$	$168kB$

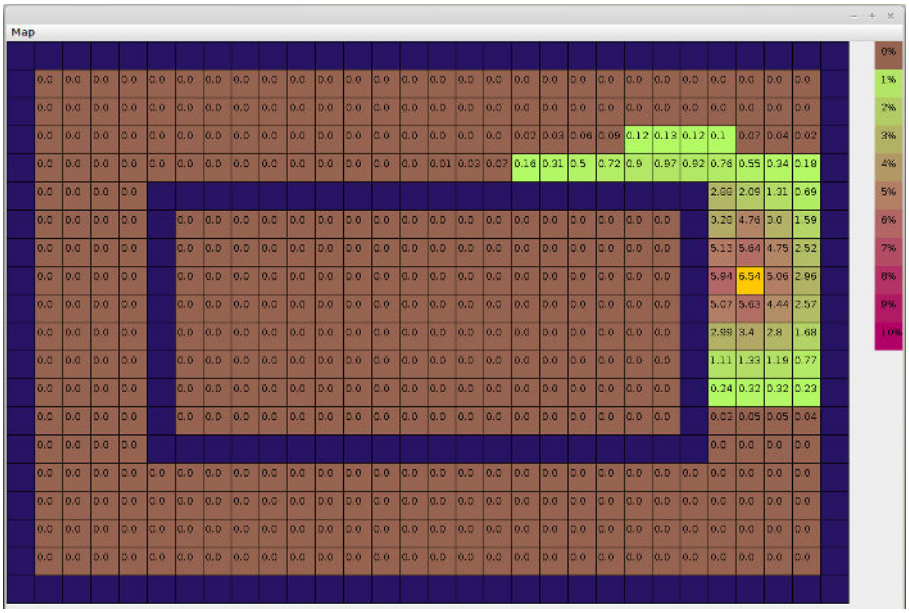
Vidíme, že sa nám podarilo efektívnou implementáciou výrazne zmenšiť veľkosť potrebnej pamäte.

Kapitola 6

Evaluácia riešenia

6.1 Pomocné nástroje na testovanie

Vytvorili sme niekoľko pomocných aplikácii, ktoré nám pomohli odhaliť silné stránky, ale aj slabiny nášho prístupu. Najprv bola vytvorená aplikácia (na počítač, nie smartfón), ktorá nám ukázala správanie algoritmov. Mapa bola manuálne vytvorená ako okruh chodieb (obrázok 6.20).

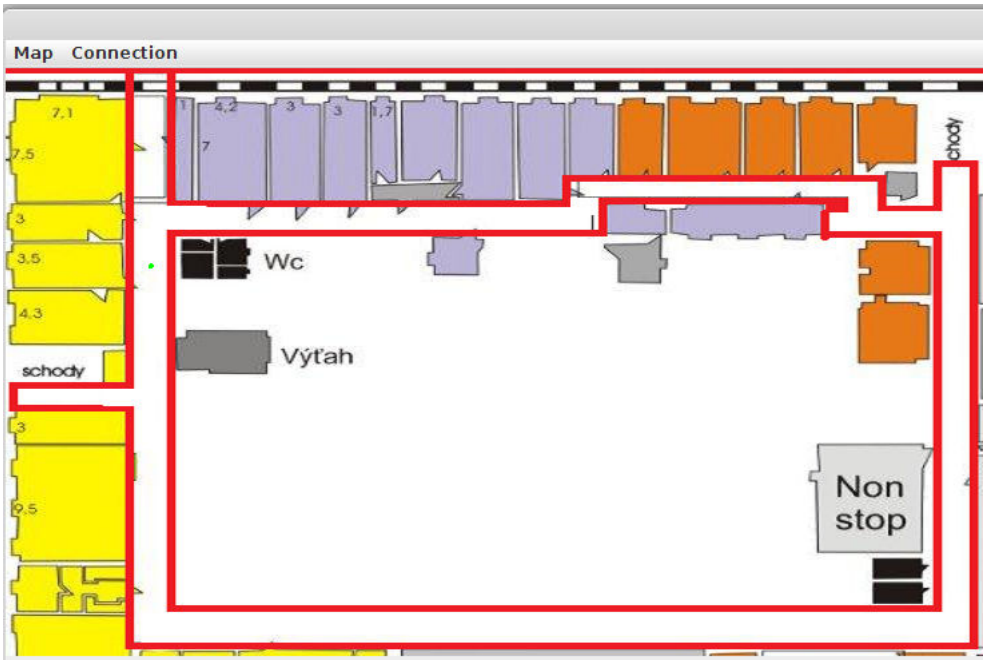


Obr. 6.20: Ukážka testovacej aplikácie.

Táto aplikácia ponúkala možnosť zobrazenia hodnôt pravdepodobností. Ovládala sa klávesnicou, kde bolo možné simulovať pohyb v ôsmych smeroch. Stlačenie klávesy znamenalo detekovanie kroku a prijatie informácie o uhle natočenia. Táto aplikácia nebola zameraná na overenie časových alebo pamäťových nárokov, ale na identifikáciu

zaujímavých vlastností algoritmu.

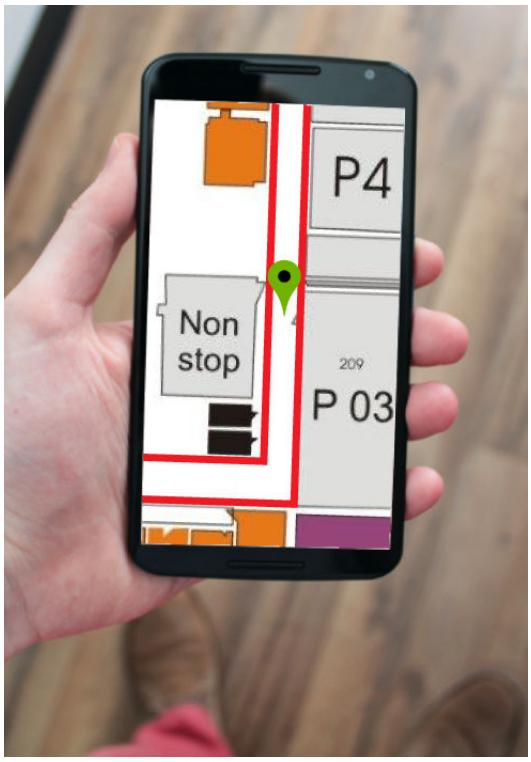
Táto aplikácia bola potom modifikovaná, aby prijímala vstup nie len z klávesnice, ale aj zo smartfónu. Vytvorili sme preto aplikáciu pre smartfón, ktorá bola schopná vytvoriť spojenie s testovacou aplikáciou cez TCP. Aplikácia na smartfóne sledovala hodnoty zo senzorov, detekovala kroky a po každom zistenom kroku posielala údaj o uhle natočenia. Testovacia aplikácia, ktorá bola vďaka WiFi sieti spojená so smartfónom, zobrazovala aktuálnu pozíciu na mape. Týmto sme overovali správanie sa algoritmu aj pri nepresných dátach zo senzorov. Vytvorili sme mapu prvého poschodia budovy PF UPJŠ, Park Angelinum 9, kde boli realizované testy.



Obr. 6.21: Ukážka testovacej aplikácie. Zelenou bodkou je označená aktuálna pozícia.

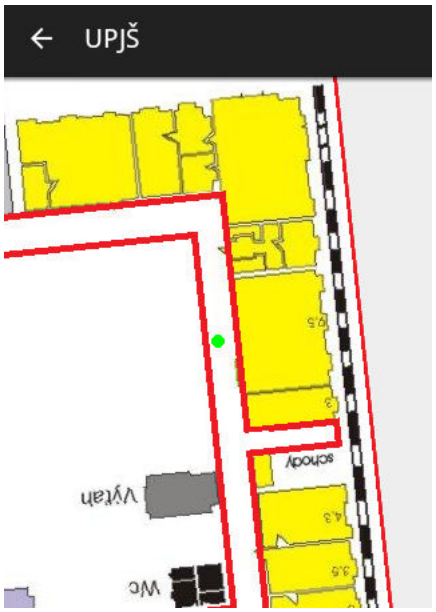
Následne bol implementovaný efektívnejší lokalizačný pravdepodobnostný algoritmus. Táto implementácia bola základom pre prototyp lokalizačnej aplikácie (obrázok 6.22).

V tomto prototype sú jednotlivé súbory pre mapu uložené v priečinkoch. Používateľovi sa zobrazí zoznam dostupných máp, z ktorých si môže vybrať aktuálnu mapu. Pridali sme aj možnosť načítania aktuálnej pozície pomocou QR kódu. V QR kóde je informácia o priečinku, kde sa nachádzajú súbory popisujúce mapu. Z tohto kódu sa tiež určí aktuálna pozícia a zmení sa príslušná hodnota v poli pravdepodobností. Používateľovi sa potom zobrazí mapa vo formáte *SVG* a jeho aktuálna pozícia. Pre potreby testovania sme pridali sledovanie pohybu používateľa. Do súboru vo formáte



Obr. 6.22: Ukážka prototypu lokalizačnej aplikácie.

CSV sa zaznamenáva pre každý detekovaný krok čas od začiatku merania, súradnice aktuálnej pozície v centimetroch a uhol natočenia získaný z kompasu. Mapa je zobrazená tak, aby používateľ videl svoju pozíciu vždy uprostred displeja a aby zobrazenie mapy zodpovedalo smeru chôdze (obrázok 6.23).



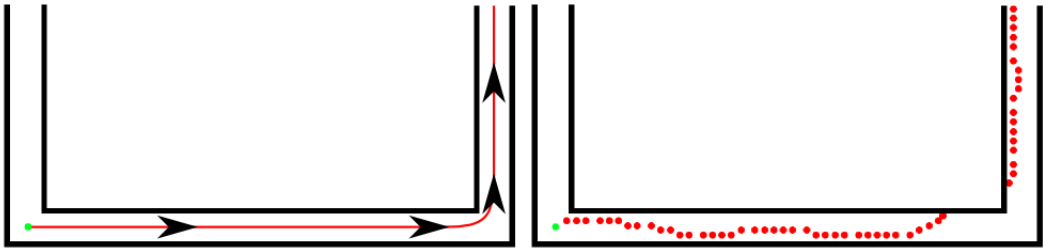
Obr. 6.23: Natočenie mapy v smere chôdze a aktuálna pozícia zarovnaná na stred displeja.

6.2 Pozorovania a presnosť lokalizácie

Prvé sledovania ukázali, že lokalizačný pravdepodobnostný model je funkčný. Vedel správne určiť aktuálnu pozíciu zo subjektívneho pohľadu používateľa. Ukázalo sa, že použitá mapa nezodpovedá reálnym rozmerom. Mapa mala rozmery 46 a 38 metrov. Skutočné rozmery sú približne o 12, resp. 8 metrov dlhšie. Pri správnych rozmeroch bola aktuálna pozícia tiež subjektívne určená správne.

Najdôležitejším pozorovaním bolo, že presnosť lokalizácie závisí od štruktúry budovy. Presnosti napomáha každé miesto, kde je používateľ nútený zmeniť smer chôdze. Pri priamočiarej chôdzi bez zmeny smeru sa kumuluje chyba spôsobená nepresným určením dĺžky kroku alebo nepresnosťami zo senzorov a algoritmu na detekciu krokov. Je to spôsobené tým, že algoritmus nemá presnú informáciu o dĺžke kroku. Z normálneho rozdelenia sa získava iba pravdepodobnosť, že dĺžka kroku je z nejakého intervalu. Križovatky chodieb alebo akékoľvek miesto, kde používateľ zmení smer chôdze, pôsobia ako synchronizačné body. Algoritmus je schopný takmer okamžite opraviť odhad aktuálnej pozície.

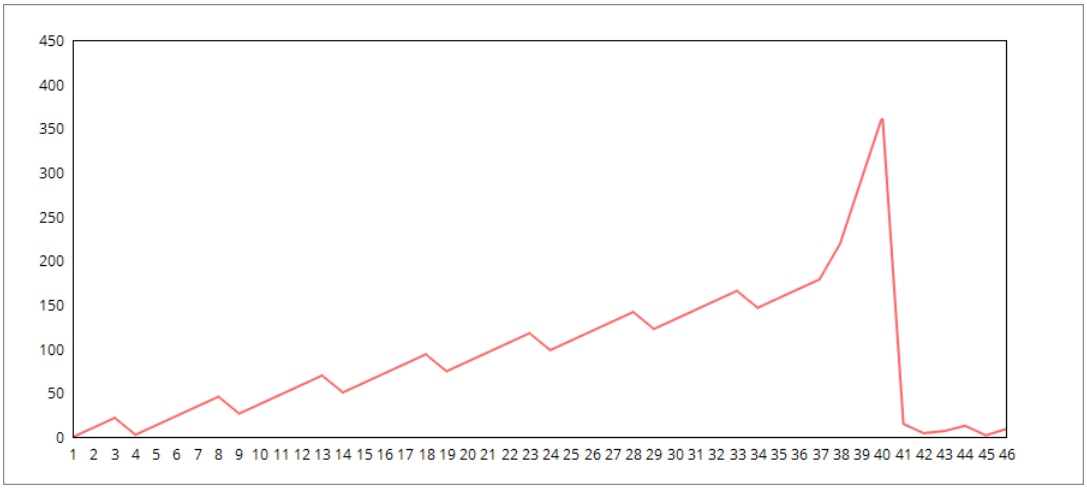
Sledovali sme situáciu na obrázku 6.24.



Obr. 6.24: Natočenie mapy v smere chôdze a aktuálna pozícia zarovnaná na stred displeja.

Používateľ kráčal chodbou dlhou 30 metrov a potom odbočil vľavo. Chyba sa kumulovala a na konci tridsaťmetrovej chodby bol rozdiel medzi detekovanou a reálnou pozíciou až 4 metre. Z pohľadu používateľa to nie je až taký problém, pretože bol schopný sa zorientovať v danej situácii, keďže kráčal rovnou chodbou. Po zmene smeru sa pozícia upravila už počas prvých dvoch krokov. Túto trasu sme absolvovali viackrát. Konkrétne hodnoty z jedného merania vidíme na obrázku 6.25.

Vo všeobecnosti sa ukazuje, že tento prístup sa vie vysporiadať s rôznou dĺžkou kroku. Ďalšou možnosťou pre zlepšenie presnosti lokalizácie by mohla byť kalibrácia dĺžky kroku. Na základe zistených údajov a prejdenej vzdialenosti by sa vypočítala priemerná dĺžka kroku pre konkrétneho používateľa. Tento prístup by vyžadoval



Obr. 6.25: Presnosť lokalizácie. Na vodorovnej osi je zobrazený počet krokov. Na zvislej osi je rozdiel medzi detekovanou pozíciou a reálnou pozíciou v centimetroch po danom počte krokov.

zmeny v implementácii, kedže masky by neboli celý čas rovnaké.

Stále ostávajú otvorené nasledovné otázky:

- Ako vplýva výber algoritmu na detekciu krokov na presnosť lokalizácie? Zmenia sa nejako slabiny lokalizačného algoritmu s použitím iného algoritmu detekcie krokov? Podľa aktuálnych pozorovaní a porovnaní výsledkov dvoch algoritmov na detekciu krokov by výber algoritmu nemal mať značný vplyv na presnosť lokalizácie.

- Ako sa algoritmus dokáže vysporiadať s rôznou dĺžkou kroku? Tu sa predpokladá otestovanie aplikácie viacerými osobami. Zo subjektívneho pohľadu sa zdá, že algoritmus je schopný vysporiadať sa s rôznou dĺžkou kroku. Pri meraniach sme menili rýchlosť chôdze a tým aj veľkosť vykonaného kroku. Nebolo vidieť žiaden veľký vplyv na presnosť lokalizácie.

- Spôsobí zmena zariadenia, na ktorom je spustená lokalizačná aplikácia, nejaké zmeny v presnosti? Má zmena držania smartfónu zásadný vplyv na lokalizáciu? Ukázalo sa, že rozdiely medzi hodnotami z kompasu z dvoch rôznych zariadení môžu byť rôzne.

Záver

Práca sa venuje oblasti indoor navigácie, ktorá sa odlišuje od bežnej outdoor navigácie tým, že nevie využiť GPS a satelitný signál. Konkrétne sa zameriava na otázku lokalizácie používateľa. Poskytujeme rozšírený prehľad existujúcich riešení s rôznym prístupom k tomuto problému. Detailnejšie je opísaný prístup s využitím akcelerometra a kompasu, ktorý určuje relatívnu pozíciu vzhľadom na zadanú navigačnú trasu.

Hlavným prínosom práce je vlastný algoritmus lokalizácie bez explicitne danej navigačnej cesty, ktorý je založený na výpočte pravdepodobností. Algoritmus prijíma na vstupe detekované kroky s hodnotou natočenia. Tento prístup ponúka aj možnosť pridania ďalších vstupných informácii, napríklad z QR kódov. Lokalizačný algoritmus bol navrhnutý tak, aby mohol byť priamou súčasťou aplikácie navigácie. Na túto skutočnosť sa myslelo aj pri spôsobe uchovávania informácii o mape. Pri návrhu algoritmu boli zohľadnené aj obmedzené výpočtové možnosti bežného smartfónu. Podstatné je, aby sa výpočet po každom detekovanom kroku vykonal v reálnom čase. Ponúkli sme návrh efektívnej implementácie nášho algoritmu. Ukázalo sa, že čas výpočtu vieme zlepšiť len minimálne, ale dokážeme obmedziť veľkosť potrebnej pamäte dosť výrazne. Tieto hodnoty závisia od veľkosti mapy.

Lokalizačný algoritmus bol implementovaný vo forme prototypu na platformu Android. Vytvorili sme tiež niekoľko podporných testovacích aplikácii, ktoré pomohli odhaliť silné stránky aj slabiny algoritmu. Evaluácia lokalizačného algoritmu ukázala, že presnosť určenia pozície používateľa závisí od štruktúry budovy. Každé miesto, kde je používateľ nútený zmeniť smer chôdze, pôsobí ako synchronizačný bod.

Náš algoritmus sa ukázal ako funkčný a použiteľný v reálnej aplikácii indoor navigácie.

Zoznam použitej literatúry

[1] RIZOS, CH., DEMPSTER, A.G., Li, B., GALLAGHER, T. Proceedings of International Conference on Indoor Positioning and Indoor Navigation (IPIN) 2012, ISBN: 978-1-4673-1954-6

[2] LINK, J.A.B., SMITH, P., VIOL, N., WEHRLE, K. FootPath: Accurate Map-based Indoor Navigation Using Smartphones. In *Proceedings of the 2011 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2011, p. 1-8.

[3] BURGARD, W., FOX, D., HENNIG, D., SCHMIDT, T. Position tracking with position probability grids. In *Advanced Mobile Robot, 1996., Proceedings of the First Euromicro Workshop on*, 1996, p. 2-9.

[4] KUUSNIEMI, H., BHUIYAN, M.Z.H. et al. Utilizing pulsed pseudolites and high-sensitivity GNSS for ubiquitous outdoor/indoor satellite navigation. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on*, 2012, p. 1-7.

[5] Indoor Navigation | Solutions powered by infsoft [online]. [cit. 2015-04-24]. Dostupné na internete: <`http://www.infsoft.com/Products/Indoor-Navigation`>

[6] Meridian | Indoor GPS, iBeacon [online]. [cit. 2015-04-24]. Dostupné na internete: <`http://meridianapps.com/`>

[7] IWASE, T., SHIBASAKI, R. Infra-free indoor positioning using only smartphone sensors. In *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, 2013, p. 1-8.

[8] RADU, V., MARINA, M.K. HiMLoc: Indoor smartphone localization via activity aware Pedestrian Dead Reckoning with selective crowdsourced WiFi fingerprin-

ting. In *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, 2013, p. 1-10.

[9] LINK, J.A.B., GERDSMEIER, F., SMITH, P., WEHRLE, K. Indoor navigation on wheels (and on foot) using smartphones. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on*, 2012, p. 1-10.

[10] LI, T., WANG, J., HUANG J. Analysis of ambiguity resolution in precise pseudolite positioning. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on*, 2012, p. 1-7.

[11] RIZOS, C., BARNES, J. et al. A new pseudolite-based positioning technology for high precision indoor and outdoor positioning. In *Int. Symp. & Exhibition on Geoinformation, Shah Alam, Malaysia*, 2003, p. 115-129.

[12] LabWerk. [online]. [cit. 2015-01-29]. Dostupné na internete: <`http://labwerk.com`>.

[13] TRUSA, F. *RSS lokalizácia v budovách s využitím WiFi signálu* : bakalárska práca. Košice: Univerzita Pavla Jozefa Šafárika, 2013. 34 s.

[14] LEDLIE, J., PARK, J., CURTIS, D. et al. *Molé: a Scalable, User-Generated WiFi Positioning Engine.* In *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2011, p. 1-10.

[15] MULLONI, A., WAGNER, D. et al. Indoor Positioning and Navigation with Camera Phones. In *Pervasive Computing, IEEE*, 2009, vol. 8, no. 2, p. 22-31.

[16] JIMENEZ, A.R., ZAMPELLA, F., SECO, F. Light-matching: A new signal of opportunity for pedestrian indoor navigation. In *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, 2013, p. 1-10.

[17] AUBECK, F., ISERT, C., GUSENBAUER, D. Camera based step detection on mobile phones, In *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2011, p. 1-7.

[18] BERNOULLI, T., DERSCH, U., KRAMMER, M. et al. Improvement of Inertial Sensor Based Indoor Navigation by Video Content Analysis. In *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2011, p. 1-9.

[19] YAN LI, WANG, J.J. A robust pedestrian navigation algorithm with low cost IMU, In *Proceedings of the Indoor Positioning and Indoor Navigation (IPIN)*, 2012, p. 1-9.

[20] Sensors overview. [online]. [cit. 2015-01-29]. Dostupné na internete: <`http://developer.android.com/guide/topics/sensors/sensors_overview.html`>.

[21] QIAN, J., MA, J. et al. An improved indoor localization method using smartphone inertial sensors. In *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, 2013, p. 1-7.

[22] OPIELA, M.: *Lokalizácia s využitím akcelerometra a kompasu* : bakalárska práca. Košice: Univerzita Pavla Jozefa Šafárika, 2013. 35 s.

[23] OpenStreetMap community: OpenStreetMap, The Free Wiki World Map [online]. [cit 2015-01-29]. Dostupné na internete: <`http://www.openstreetmap.org`>.

[24] FOX, D., HIGHTOWER, J., LIAO, L. et al. Bayesian filtering for location estimation. In *Pervasive Computing, IEEE*, 2003, vol. 2, no. 3, p 24-33.

[25] GALOV, A., MOSCHEVIKIN, A. Bayesian filters for ToF and RSS measurements for indoor positioning of a mobile object. In *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, 2013, p. 1-8.

[26] GUSENBAUER, D., ISERT, C., KRÖSCHE, J. Self-contained indoor positioning on off-the-shelf mobile devices. In *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*, 2010, p. 1-9.

[27] J. KRUMM, L. WILLIAMS, and G. SMITH. SmartMoveX on a graph – an inexpensive active badge tracker. In *UbiComp '02 Proceedings of the 4th international conference on Ubiquitous Computing*, 2002, p. 299-307.