České vysoké učení technické v Praze

Fakulta elektrotechnická

Kybernetika a robotika

Bakalářská práce

Lokalizace osob v budově s využitím bezdrátové technologie

Jan Trejbal

Květen 2015

Vedoucí práce: Ing. Karel Košnar, Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Jan T r e j b a l

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Lokalizace osob v budově s využitím bezdrátové technologie

Pokyny pro vypracování:

1. Seznamte se s metodami lokalizace za přítomnosti neurčitosti.
2. Naimplementujte vybranou metodu využívající skryté markovovské modely.
3. Proveďte experimentální ověření.
4. Vyhodnoťte kvalitu implementované metody.

Seznam odborné literatury:

[1] Zahid F., Rosdiadee N., and Mahamod I., “Recent Advances in Wireless Indoor Localization Techniques and System,” Journal of Computer Networks and Communications, vol. 2013, Article ID 185138, 12 pages, 2013. doi:10.1155/2013/185138
[2] Arthi, R.; Murugan, K., "Localization in Wireless Sensor Networks by Hidden Markov Model," Advanced Computing (ICoAC), 2010 Second International Conference on , vol., no., pp.14,18, 14-16 Dec. 2010 doi: 10.1109/ICOAC.2010.5725355
[3] Ibrahim, M., and Moustafa Y.. "A hidden markov model for localization using low-end gsm cell phones."Communications (ICC), 2011 IEEE International Conference on. IEEE, 2011.

Vedoucí bakalářské práce: Ing. Karel Košnar, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 20. 1. 2015

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 20. 5. 2015

Poděkování

Rád bych poděkoval Ing. Karlu Košnarovi, Ph.D. za vedení této bakalářské práce a za cenné rady, díky kterým jsem zdárně dokončil tuto bakalářskou práci.

Rád bych též poděkoval rodině za podporu a trpělivost po dobu mého studia.

Abstrakt

Tato bakalářská práce se zabývá lokalizací osob v budově s využitím bezdrátové technologie. Lokalizace je řešena s využitím skrytých Markovských modelů a porovnávána je s k-NN klasifikací. Pro lokalizaci bylo v rámci práce napsáno několik programů a aplikací v jazyku C++ a PHP. V závěru je porovnáno několik konfigurací aplikace a vyhodnocena nejvhodnější konfigurace.

Klíčová slova: Skryté Markovské modely, lokalizace, matlab, C++, PHP, relační databáze

Abstract

This thesis deals with the localization of people in buildings using wireless technology. Localization is solved using hidden Markov models and compares them with the k-NN classification. I wrote several programs and applications which were written in C++ and PHP. I compared several configuratios of applications and I evaluated the most appropriate configuration.

Keywords: Hidden Markov models, localization, matlab, C++, PHP, relational database

Obsah

Seznam obrázků

Kapitola 1
Úvod

1.1 Definice úlohy

Cílem mé bakalářské práce je lokalizovat osoby v budově s přesností na místnosti s využitím v budově rozmístěného hardwaru. Mezi rozmístěný hardware patří přijímače a lokalizovanou osobou nesený vysílač. Lokalizovat osobu v budově je nesnadné, především díky interferencím vznikajícím průchodem signálu z nesených vysílačů skrz stěny k přijímačům.

Tento hardware lze případně zaměnit za WiFi access pointy a mobilní telefony na ně připojené. Díky čemu lze snížit pořizovací náklady využitím existující infrastruktury.

1.2 Metody lokalizace

Mezi uvažované metody lokalizace, či klasifikace pro tuto bakalářskou práci patří např. triangulace, k-NN klasifikace a skryté Markovské modely.

1.2.1 Triangulace

Lokalizace pomocí triangulace vychází z geometrie a z vlastností trojúhelníku. V této metodě se využívá znalosti vzdáleností mezi známými body, mezi známými body změříme úhly v bodě u kterého chceme určit polohu, z těchto úhlů lze vypočítat jednoduchou matematikou poloha, pro ilustraci nákres metody na obrázku 1, na kterém je zachycena triangulace pro stanovení vzdálenosti lodi od pobřeží v které po změření úhlů stačí využít vzorec:

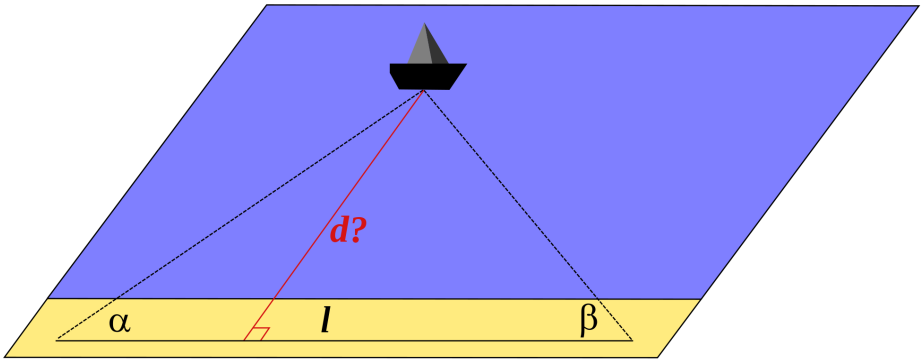
$$d = l * sin(\alpha) * sin(\beta)$$

Pro lokalizaci osob v budově tato metoda není vhodná. Měření úhlů mezi známými body v budově je velice špatně realizovatelné. Použití sil signálů přepočtených na vzdálenost by použití metody umožnila, tato metoda by však špatně reagovala na různě tlusté, nebo mokré stěny, které ovlivňují signál jimi prostupující.

1.2.2 K-NN

Lokalizace (klasifikace) pomocí k-NN[1] [4] je běžně užívanou metodou, díky její jednoduchosti. V této metodě se určí nejvhodnější kandidát pomocí nejmenší vzdálenosti od

[1] Nearest Neighbour classification: Klasifikace podle nejbližších sousedů

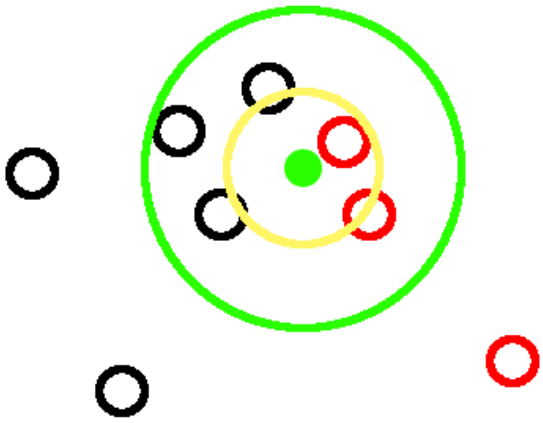


Obrázek 1: Ukázka použití triangulace [1]

'k' prvků. Z tohoto postupu plyne nutnost existence metody vyčíslující vzdálenost mezi jednotlivými prvky.

Náhled vyhodnocování pomocí k-NN je na obrázku 2, na kterém zelené kolečko znázorňuje prvek, který chceme ohodnotit, červené a černé kruhy znázorňují referenční vzorky. Při použití 1-NN znázorňuje vyhodnocení žlutý kruh, jako ohodnocení by byl zvolen červený kruh. Použití 3-NN znázorněné zelený kruhem oproti 1-NN však ohodnotí prvek jako černý.

Tuto metodu lze využít, bohužel však tato metoda nezohledňuje reálný pohyb osob budovou. Umožňuje tak například procházení zdí budov, nebo skoky mezi patry bez použití schodiště.



Obrázek 2: Ukázka vyhodnocení pomocí k-NN

Kapitola 2
Skryté Markovské modely

Skryté Markovské modely (HMM) [2] jsou rozšířením Markovských modelů [5], v kterých řetěz stavů není přímo pozorovatelný.

Oproti například triangulaci zmíněné v sekci 1.2.1 a K-NN ze sekce 1.2.2 má HMM výhodu ve specifikování možných přechodů mezi stavy (v našem případě místnosti), respektive v možnosti jejich zamezení viz obrázek 3 (tento graf lze popsat maticí tranzitivity) z kterého je zřejmé, že neexistuje přechod mezi π_1 a π_6, ale že existuje přechod z π_1 do π_2 nebo π_5. Definováním přechodů mezi stavy, tak lze zamezit chybám způsobeným interferencí a následným špatným určením stavu. Tyto chyby jsou automaticky eliminovány díky absenci přímého přechodu mezi stavy (např. pro přechod z místnosti do místnosti je nutné využít chodbu).

Jelikož nemáme přímou informaci o aktuálním stavu (místnosti) nad přechody mezi stavy vybudujeme další matici (matice emisivity), která udává pravděpodobnost, s jakou daný stav odpovídá vektoru sil signálů z přijímačů.

Matematický model HMM pro náš případ můžeme reprezentovat pomocí S, V, T, E a O, kde:

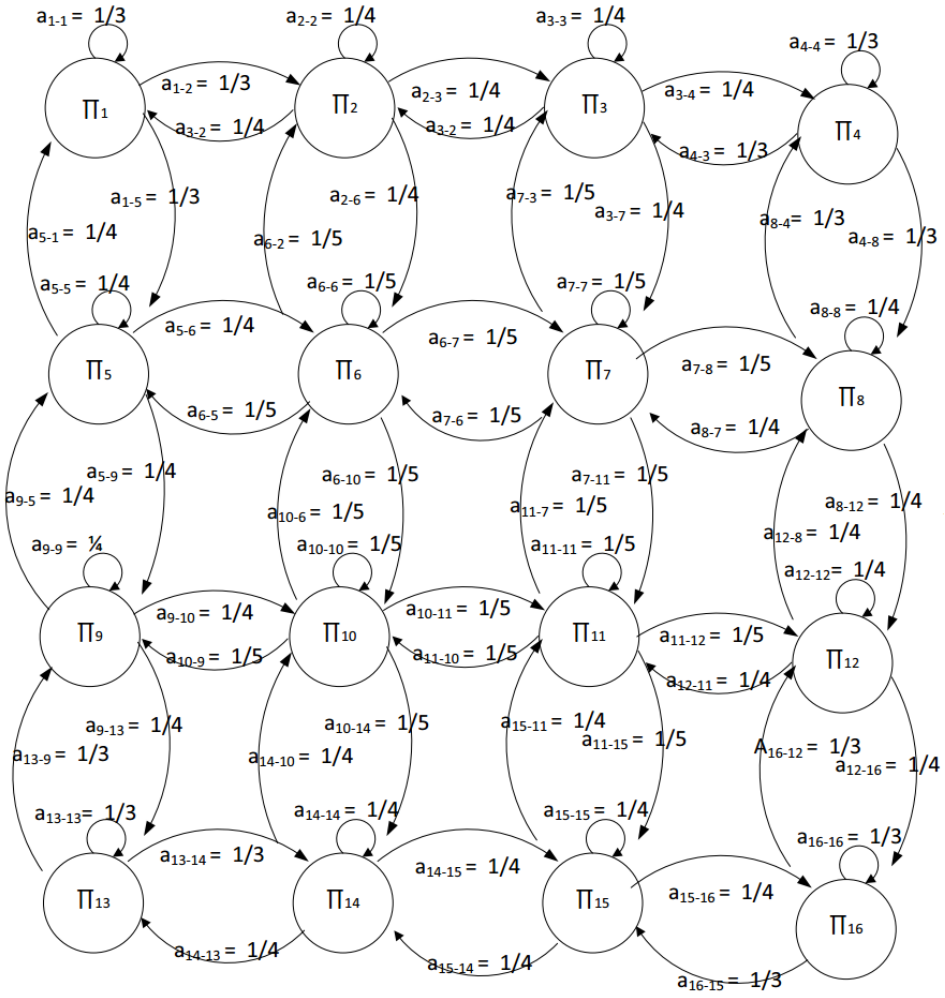
- $S = \{S_1, S_2, ..., S_N\}$ - S_i jsou jednotlivé místnosti a $N = |S|$ v S jsou tedy všechny místnosti,
- $V = \{v_1, v_2, ..., v_M\}$ - v_i jsou jednotlivá pozorování (upravené síly signálů) a $M = |V|$ ve V jsou tedy všechna získaná data,
- $T = \{t_{ij}\}$ je matice tranzitivit v které každý prvek t_{ij} odpovídá pravděpodobnosti přechodu z S_i do S_j,
- $E = \{e_{ij}\}$ je matice emisivit, v které $i < N$ a $j < Q$ kde Q je počet všech možných pozorování, e_{ij} pak odpovídá pravděpodobnosti emise ze stavu do daného pozorování a
- $O = \{o_1, o_2, ..., o_M\}$ je pak hledaná sekvence místností.

2.1 Trénování modelu

Pro trénování modelu jsem si vybral interativní metodu [6]. Trénování pomocí této metody lze zjednodušeně popsat těmito kroky:

1. Odhad počátečních pravděpodobností - v základu rovnoměrné.

[2] Hidden Markov Model: Skryté Markovské modely



Obrázek 3: Ukázkové schéma tranzitivit HMM [2]

2. Na základě emisivity a tranzitivity určení nejpravděpodobnější sekvence stavů z vektorů sil signálů.

3. Odhad nových pravděpodobností na základě očekávaných a vypočtených sekvencí stavů.

4. Provádíme n iterací 2. a 3. bodu algoritmu do doby, než jsme spokojeni s odchylkou ϵ.

Po dokončení trénování máme připraven model pro rozpoznávání, tj. máme naplněny S, T a E ze sekce 1.2.2.

Pro úplnost zmíním existenci Baumova-Welchova algoritmu [7], ten jsem však v práci nevyužil, lze ho však např. využít pro další zpřesňování modelu.

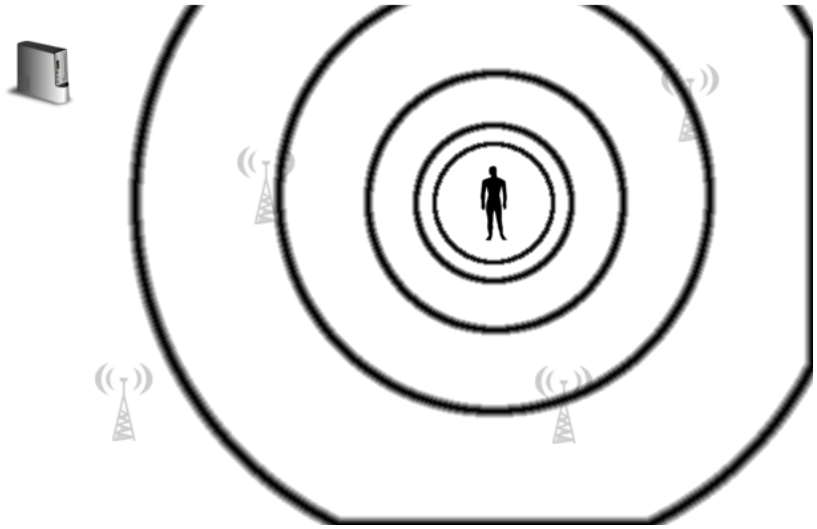
2.2 Použití modelu

Pro nalezení nejpravděpodobnější cesty na základě vektorů sil signálů a natrénovaného modelu jsem se rozhodl využít Viterbiho algoritmus [8], který je vhodným algoritmem, pro určení nejpravděpodobnější sekvence stavů. Tento algoritmus se využívá například v GSM, CDMA, satelitní komunikaci a dalších systémech využívajících digitální kódování [9].

Do tohoto algoritmu vstupují data připravená z trénování, tj. S, T a E, dále pak seznam pozorování V, výstupem je pak hledaná sekvence O.

Kapitola 3
Jednotky IMA

Pro pokusy mi byl umožněn přístup k jednotkám od firmy IMA[3], které obsahují řídící jednotku, přijímače a vysílače. Nákres struktury hardwaru je na obrázku 4.



Obrázek 4: Nákres struktury IMA

3.1 Vysílače

Vysílače jsou přenosné, zhruba o velikosti pageru. Na svém těle mají umístěné tlačítko, po jeho stisku dojde k neprodlenému vyslání signálu z vysílače. Vysílač je k nahlédnutí na obrázku 5.

Vysílače vysílají signál v pravidelných intervalech ihned po zapnutí. Jeden z vysílačů byl v době testování nastaven na pravidelné vysílání signálu jednou za 1s, druhý 4x za stejnou dobu.

Signál je vysílaný v pásmu 2,4GHz (tedy volné pásmo).

3.2 Přijímače

Přijímače jsou nepřenosné, o velikosti běžného notebookového zdroje. Na svém těle mají dva SMA konektory pro antény, konektor pro napájení, vypínač a diodu indikující

[3]Institut Mikroelektronických Aplikací

Obrázek 5: Vysílač IMA [3]

zapnutí přijímače. Přijímač je k nahlédnutí na obrázku 6. Tyto přijímače se automaticky spojí s řídící jednotkou, které předávají sebraná data.

3.3 Řídící jednotka

Řídící jednotka je mikro PC, které má na svém těle napájecí konektor, síťový konektor (RJ45) a USB konektor. Do USB konektoru je zapojen ZigBee přijímač, který sbírá data zaslaná z přijímačů. Tato jednotka přijatá data odesílá jako proud dat.

3.4 Zpracování dat

Zpracování dat probíhá v této sekvenci:

- Vysílač vyšle data (automaticky, nebo po stisku tlačítka),
- všechny přijímače v dosahu přijmou vyslaný signál a ohodnotí jeho úroveň,
- řídící jednotka od přijímačů obdrží informaci o vysílači, identifikátor vyslaného signálu (umožňuje spárovat přijatá data z jednotlivých přijímačů) a sílu signálu,
- řídící jednotka vysílá přijatá data jako proud dat na TCP/IP portu 1112. Obsah proudu dat je k nahlédnutí na obrázku 7. Na tomto obrázku jsou vidět servisní informace (označené '#' v druhém sloupci), ty jsou pro účely měření nezajímavé.

Obrázek 6: Přijímač IMA [3]

Zajímavé jsou data která mají v druhém sloupci 'D' v daném řádku jsou obsažená data jako: identifikátor přijímače (4. sloupec), identifikátor vysílače (5. sloupec), identifikátor vyslaného signálu (6. sloupec), sílu signálu (7. sloupec) a informaci o stisku tlačítka (8. sloupec). Všechna takto vyslaná data jsou v šestnáctkové soustavě.

```
1218192519.121949 # NWK: Routing (00)
1218192519.125206 # NWK: Dcfm
1218192519.12875  # NWK:   success
1218192519.131967 # NWK:   route
1218192519.328195 # NWK: Data Ind (009D, 0C)
1218192519.332687 # NWK:   filter passed
1218192519.337211 D LQ 009C 004A FC0E 78 0
1218192519.344248 # NWK: Data Ind (00A3, 41)
1218192519.348555 # NWK:   filter passed
1218192519.352896 D LQ 00A4 004A FC0E 8A 0
1218192520.325301 # NWK: Data Ind (009D, 0D)
1218192520.3299   # NWK:   filter passed
1218192520.334422 D LQ 009C 004A A406 78 0
1218192520.340816 # NWK: Data Ind (00A3, 42)
1218192520.345064 # NWK:   filter passed
```

Obrázek 7: Proud dat z řídící jednotky

Kapitola 4
Implementace

Pro potřeby bakalářské práce jsem implementoval programy v jazyce C++ a aplikaci v PHP. Prvním programem byl program pro sběr dat, který byl vyvíjen zároveň s aplikací v PHP, které umožňuje vlastní sběr dat. Druhý implementovaný program zpracovává sebraná data, z kterých trénuje model a následně rozpoznává nejpravděpodobnější sekvenci navštívených místností, kterou porovnává s referenční metodou k-NN.

4.1 Sběr dat

Sběr dat má aplikace provádět připojením k řídící jednotce IMA popsané v sekci 3.3, následně má tato data přeparsovat a uložit pro další použití. Od toho se odvíjí její struktura:

4.1.1 Ukládání informací v průběhu měření

Program pro sběr dat[4] je psán v jazyce C++ s maximální snahou využívat přednosti OOP[5], jako jsou například rozhraní (interface)[6] umožňující v navrženém programu například snadno změnit SQL úložiště za úložiště zcela jiného typu.

Základní funkce programu jsou:

- Připojení k úložišti,
- připojení k řídící jednotce IMA 3.3,
- parsování proudu dat z řídící jednotky a
- uložení výsledků do úložiště

které jsou popsány v následujících sekcích:

4.1.1.1 Připojení k úložišti

Jako úložiště jsem zvolil relační databázi MySQL [10], konkrétně její odnož MariaDB [11], která je oproti MySQL od Oracle efektivnější. Databázi jsem zvolil InnoDB [12], která mimo jiné umožňuje využití transakčních operací.

Pro tuto databázi jsem implementoval SqlStorage jako rozhraní IStorage, ve kterém jsem využil existující C++ knihovny pro MySQL[13]. V této implementaci jsem použil

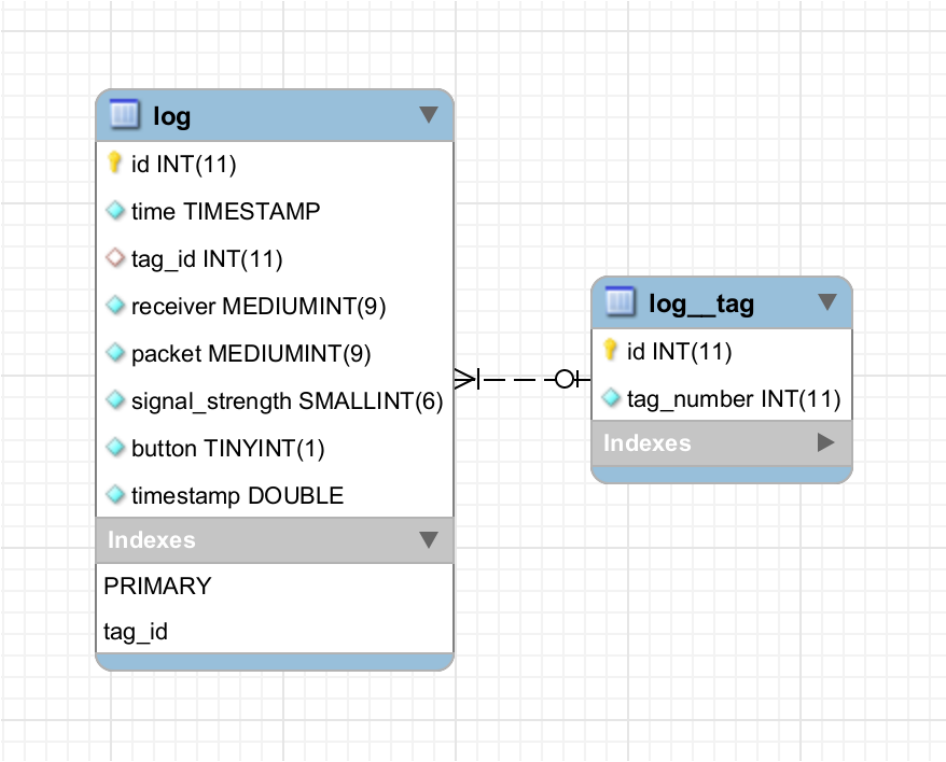
[4]K dispozici na přiloženém CD a online na https://bitbucket.org/trejjam/ima

[5]Object-oriented programming: Objektově orientované programování

[6]V jazyce C++ nejsou definované rozhraní (interface), jako např. v PHP(PHP: Hypertext Preprocessor), tuto absenci obcházím pomocí neimlementované (tj. pouze definované) třídy

předpřipravené dotazy (prepared statements [14]), které mimo zlepšení časových prodlev přináší i typovou kontrolu, případně úpravu dat.

Vzájemné vztahy tabulek databáze využívaných touto implementací jsou vyobrazeny pomocí ER[7] modelu na obrázku 8. Tento model popisuje jednoduchou relaci mezi tabulkou se sebranými daty a vysílačem z kterého byla tato data odeslána.



Obrázek 8: ER diagram

4.1.1.2 Připojení k řídící jednotce IMA

Připojení k řídící jednotce IMA se uskuteční na základě zadání IP adresy a portu a následném navázání TCP/IP spojení k jednotce. Řídící jednotka takto vytvořeným spojením zasílá do aplikace jednoduchý proud dat popsaným v sekci 3.4.

4.1.1.3 Parsování proudu dat z řídící jednotky

Proud dat z řídící jednotky 4.1.1.2 vstupuje do parseru, který má za úkol vybrat z dat užitečné informace (kombinace času, ID vysílače, ID přijímače, ...). Prvním krokem je rozdělení dat podle oddělovače (v tomto případě je to znak '\n' - odřádkování). Dalším krokem je identifikace datových řádků, ty jsou rozpoznatelné podle klíčového znaku 'D' v druhém sloupci. Posledním krokem parsování je převod jednotlivých informací na datové typy vhodné pro uchování a další zpracování. Tyto datové typy jsem určil následovně:

[7] Entity-relationship model: entitně vztahový model

- čas - double
- ID přijímače - int
- ID vysílače - int
- ID měření - int
- síla signálu - int
- stisknuté tlačítko - int[8]

Informace získané z parseru jsou předávány do popsané implementace rozhraní IStorage (tj. do SqlStorage) popsané v sekci 4.1.1.1, které je následně ukládá do databáze pro pozdější využití.

Zdrojové kódy jsou spravovány GIT repositářem[9], použitým především pro možnost verzování, díky jeho existenci je snadné spolupracovat na projektu s více účastníky. Ostatní zdrojové kódy a texty jsou také verzované z podobných důvodů jako tento program.

4.1.2 Definování fyzického prostředí

Pro přiřazení naměřených dat k reálným prostorám jsem vytvořil jednoduchou webovou aplikaci [10]. Při jejímž vytváření bylo myšleno na snadné vložení map budov a následné specifikování místností, též na případné budoucí zobrazování pozice vysílače (lokalizování osoby).

Aplikace je naprogramována v jazyce PHP s napojením na stejnou MariaDB databázi jako aplikace popsaná v sekci 4.1.1, díky tomuto propojení lze snadno přiřazovat naměřená data k jednotlivým místnostem.

Jádrem aplikace je Nette Framework 2.3 [15] s moduly (extensions) dostupnými přes Composer[11]:

- trejjam/authorization [16] - správa uživatelů a jejich rolí s napojením na relační databázi pro pohodlné nastavování práv z administrace aplikace
- kdyby/console [17] - knihovna pro snadné vytváření konzolových příkazů v PHP pro Nette Framework

Při psaní aplikace byly využity principy jako DI[12], DRY[13]. Pro automatizování při vývoji je použit Grunt[14]. Nasazování aplikace je řešeno pomocí verzovacího nástroje GIT a vzdáleného repositáře.

[8] pro tlačítko by byl vhodnější boolean, avšak tento typ má slabou podporu v některých databázových systémech, proto jsem zvolil int

[9] K dispozici na přiloženém CD a online na https://bitbucket.org/trejjam/ima

[10] K dispozici na přiloženém CD a online na https://bitbucket.org/trejjam/ima-web

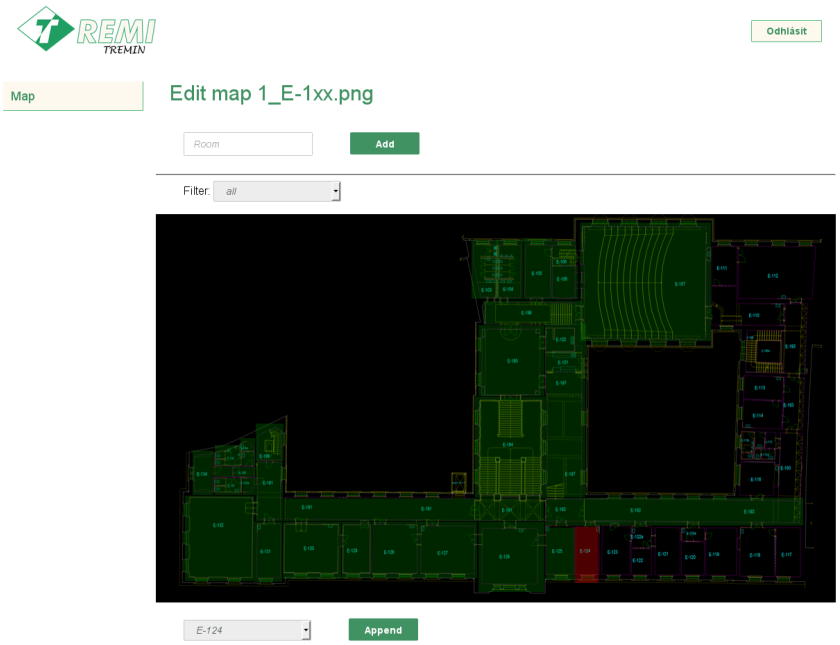
[11] Balíkovací systém pro PHP

[12] Dependency Injection

[13] Don't repeat yourself

[14] http://gruntjs.com/

Aplikace umožňuje nahrání obrázku mapy pomocí standardního souborového dialogu. Na nahrané mapě následovně umožňuje specifikování jmen místností. Místnosti, které existují v databázi je možno namapovat na mapu pomocí gesta drag & drop. Náhled aplikace je na obrázku 9, nastavené místnosti jsou označené zelenou barvou, právě mapovaná místnost je označena červeně.



Obrázek 9: Nastavení mapy

Připravená mapa se specifikovanými místnostmi je využívána při sběru dat. A to tak, že umožňuje uživateli provádějícímu měření jednoznačně určit, v které místnosti se nachází. Toto uživatel provede například z mobilního telefonu, s použitím jeho webového prohlížeče, kde specifikuje nesený vysílač (tag) a místnost v které se nachází. Náhled použité aplikace je na obrázku 10.

Získáním těchto informací je schopen program popsaný v sekci 4.1.1 spojit příchozí data s místností. Toto spojení umožňuje budoucí operace s daty, například trénování modelu.

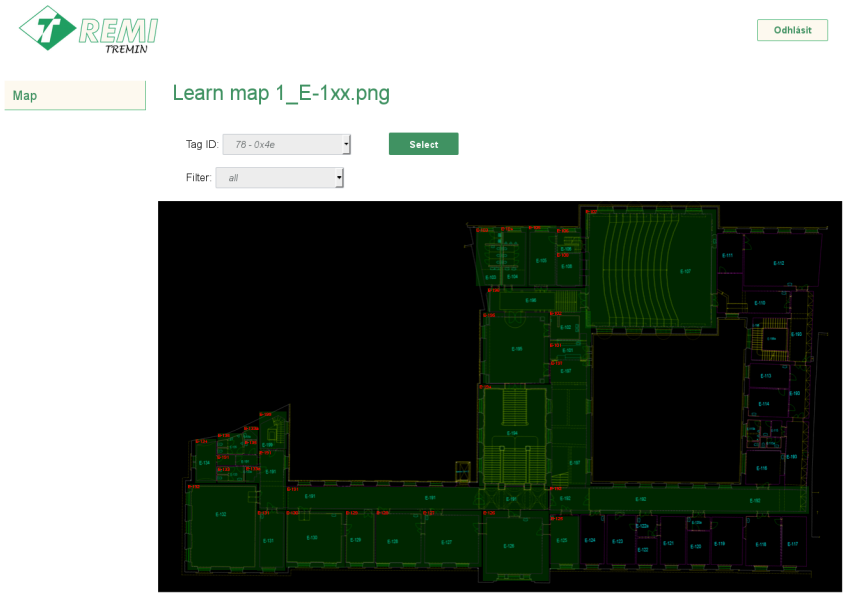
4.2 Učení a rozpoznávání ze sebraných dat

Pro učení jsem připravil specializovanou aplikaci, která využívá data připravená aplikací popsanou v sekci 4.1.1. Tato aplikace[15] je též psaná v jazyce C++.

Základní funkcionality aplikace jsou:

- Připojení k úložišti,

[15] K dispozici na přiloženém CD a online na https://bitbucket.org/trejjam/imacomputing



Obrázek 10: Aplikace používaná při učení

- připojení k Matlabu,
- připravení dat pro trénování a rozpoznávání,
- vystavění modelu z dat,
- rozpoznání nejpravděpodobnější sekvence stavů z dat a natrénovaného modelu a
- export výsledků.

4.2.1 Připojení k úložišti

Připojení k úložišti využívá stejnou knihovnu jako program pro sběr dat 4.1.1. Jen s tím rozdílem, že do úložiště nezapisuje.

Po úspěšném připojení k úložišti je třeba vybrat měření které bude použito pro trénování a rozpoznávání. Toho je docíleno pomocí klasického řádkového vstupu, pomocí kterého lze zvolit jedno a více měření.

4.2.2 Připojení k Matlabu

Pro připojení do programu Matlab jsem využil poskytovanou knihovnu [18], která umožňuje volat vlastní příkazy automaticky z vlastní aplikace, jako kdyby byli ručně zadávané do konzole Matlabu. Původně jsem chtěl využít tohoto API [16] i pro přenos dat mezi aplikací a Matlabem, ale zde nejspíše kvůli neoptimálnímu nadefinování typů polí

[16] Application Programming Interface: soubor funkcí, procedur, ... pro komunikaci s aplikací

docházelo k různý změnám v datech. Proto jsem nakonec sáhl po ne zcela optimálním řešení a to přenášet data přes prostředníka, kterým se stalo několik textových souborů.

4.2.3 Připravení dat pro trénování a rozpoznávání

Sebraná data nejsou vhodná pro přímé použití v HMM, proto jsem data nejdříve pročistil, kvantoval a překódoval, tyto úpravy jsou popsány v následujících sekcích:

4.2.3.1 Pročištění dat

Aby nevznikl přeučený model (tj. model, který díky velkému množství dat zpřesní své hodnoty tak, že při nasazení není schopen správně zpracovat/zařadit většinu dat) rozhodl jsem se sebraná data pročistit. Metodu čištění jsem zvolil tak, že používám data s rozestupy časových značek od předchozí i následující časové značky naměřených údajů.

Tedy pro každý použitý záznam platí (s - časová značka záznamu; n - počet záznamů; i = 0, 1, (n-2); k = zvolený rozestup):

$$s_i <= (s_{i+1} + k) <= (s_{i+2} + k)$$

4.2.3.2 Kvantování dat

Protože jsou data z řídící jednotky IMA 3.3 spojitá (s intervalem 0-255), rozhodl jsem se tato data pro snazší implementaci HMM 1.2.2 kvantovat. Kvantily jsem stanovil objektivně z naměřených dat a to pomocí vzorce (r - rozsah, q - počet kvantilů, d - přijatá data, round - zaokrouhlení dolů, o - přiřazený kvantil):

$$o = round(d/(r/q))$$

4.2.3.3 Překódování dat

Základní použití HMM je omezeno na skalární hodnoty, proto bylo třeba vektory sil signálů zobrazit na skalární hodnotu. Pro toto zobrazení (překódování) jsem si stanovil podmínku, aby bylo prosté, tj. dva různé vektory se nesmí zobrazit do stejného prvku.

Překódování jsem implementoval jako jednoduchý přepočet z n-kové soustavy (kde n je počet kvantilů) do desítkové soustavy. Vezmeme-li si náhodný vzorek ze čtyř přijímačů o pěti kvantilech implementovaný přepočet vypadá následovně:

$$0143_{(5)} = 0 * 5^3 + 1 * 5^2 + 4 * 5^1 + 3 * 5^0 = 48_{(10)}$$

4.2.4 Vystavění modelu z dat

Při implementování trénování jsem využil toho, že iterativní algoritmus 2.1 je implementovaný v Matlabu [19]. Jeho základní volání vypadá takto:

$$[TRANS, EMIS] = hmmestimate(seq, states)$$

Vstup do tohoto algoritmu jsou pročištěná, kvantovaná a překódovaná data (postup zmíněn v sekcích 4.2.3.1, 4.2.3.2 a 4.2.3.3). Výstupem jsou pak hledané matice emisivity a tranzitivity.

Pro použití na testovaných datech bylo třeba využít i volitelné parametry funkce:

Prvním parametrem je počet symbolů, protože pročištěná data mohou být ochuzena o některé symboly oproti datům určeným k rozpoznávání, což při případné absenci a následném rozpoznávání skončí chybou.

Druhým parametrem je předpřipravená matice tranzitivit, kterou jsem předvyplnil jedničkami na diagonále (tj. 100% pravděpodobnost přechodu do stejného stavu) a to z důvodu velice podobného jako u prvního parametru. Zde však s vazbou na možnost chybějících stavů, které způsobí nulový řádek výsledné matice, což vede k chybě při rozpoznávání.

Třetím a posledním parametrem je předpřipravená matice emisivit, kterou jsem předvyplnil tak aby emisivity ze všech stavů byly stejné (tj. všechny prvky matice mají hodnotu 1/počet symbolů).

4.2.5 Rozpoznání nejpravděpodobnější sekvence stavů z dat a natrénovaného modelu

Pro rozpoznávání Viterbiho algoritmem popsaným v sekci 2.2 jsem využil funkce implementované v Matlabu [20]. Jejíž volání vypadá takto:

$$STATES = hmmviterbi(seq, TRANS, EMIS)$$

Do této funkce předávám pročištěná, kvantovaná a překódovaná data (postup zmíněn v sekcích 4.2.3.1, 4.2.3.2 a 4.2.3.3), s jedinou úpravou oproti trénování, kterou je změna zdrojového vysílače. Výstupem funkce je nejpravděpodobnější řetězec stavů (místností), kterými se vysílač pohyboval.

4.3 Jiná metoda učení a rozpoznávání

Jako referenční algoritmus jsem vybral k-NN 1.2.2, který je považován za dobrou referenční metodu klasifikace. Pro tuto metodu jsem implementoval trénování a rozpoznávání s využitím algoritmů dostupných v Matlabu.

4.4 Export výsledků

Výsledky porovnání úspěšnosti rozpoznání HMM jejichž algoritmus je popsán v sekci 4.2.5 a KNN v sekci 4.3 se ukládají do jednoduchého souboru ve formátu CSV[17], který umožňuje jejich snadné budoucí použití.

[17] Comma-separated values: hodnoty oddělené čárkami

Kapitola 5
Experimenty

Při provádění experimentů jsem využil napsané aplikace a znalosti síťové komunikace, které přišli vhod např. při zpřístupňování zařízení umístěné za NATem[18] ze sítě Internet

5.1 Sběr dat

Sběr dat jsme prováděli v budově E ČVUT FEL na Karlově náměstí v bloku budovy směřující k budově G. Přijímače popsané v sekci 3.2 jsme rozmístili v 1. patře viz obrázek 11.

Obrázek 11: Mapa s vyznačenými přijímači

Před vlastním měřením bylo nutné spojit, či umožnit spojení všech zařízení, tj. jednotky IMA, sběrné zařízení (notebook) a mobilní telefony.

Stabilní přístup mobilních telefonů k notebooku, na kterém běží server s aplikací, popsané v sekci 4.1.2, umožňující specifikování místnosti, v které se subjekt provádějící měření nachází, jsme vyřešili, pomocí vlastního soukromého serveru s veřejnou IP, z kterého jsme vytvořili reverzní tunel přes SSH [19], díky čemu byl notebook dostupný, i v případě, kdy mobilní telefony přešli na mobilní připojení (z důvodu nedokonalého wifi pokrytí) a nebyl-li tedy mobilní telefon přímo ve školní síti.

Jednotky IMA popsané v kapitole 2.2 jsme s notebookem určeným pro běh aplikace připravené pro sběr dat, popsané v sekci 4.1.1, propojily pomocí routeru nakon-

[18] Network Address Translation: překlad síťových adres
[19] SSH: Secure Shell

figurovaného jako switch, který jsme připojili do školní sítě, abychom zajistili stabilní připojení k zmíněnému veřejnému serveru. Do tohoto switche jsme připojili notebook, kterým jsme proskenovali zařízení viditelná v síti, následně jsme připojili a spustili řídící jednotku IMA, znovu jsme proskenovali síť a z rozdílu viditelných zařízení jsme identifikovali IP adresu řídící jednotky IMA, která je jedním z parametrů aplikace pro sběr dat.

Propojením jednotlivých zařízení a spuštěním aplikací jsme zajistili vše potřebné k sběru dat, který probíhal tak, že jsme v mobilním zařízení specifikovali nesený vysílač (tag) a při přechodu z místnosti do místnosti jsme tuto místnost označovali ve webové aplikaci. Informace o místnosti z této aplikace měla k dispozici aplikace zpracovávající data z řídící jednotky IMA, která je přiřazovala k přijímaným datům z jednotky, čímž jsme získali sekvenci vektorů sil signálů s informací ke které místnosti tato data patří.

Sběry dat jsme prováděli s kolegou Richardem Bláhou, který má pro sebraná data využití v rámci vlastní bakalářské práce. Sběr jsme prováděli současně s tím, že každý z nás nesl jeden vysílač a pohyboval se nezávisle na druhém. V prvním měření jsme naměřili přibližně 11500 záznamů, v druhém měření 34000. Při druhém měření jsme rozšířili sběr dat i na druhé patro bez změny rozmístění přijímačů. Abychom zamezili vlastnímu vlivu na měření při druhém měření jsme si vysílače s kolegou vyměnili.

5.2 Trénování a rozpoznávání z dat

Trénování a rozpoznávání nad sebranými daty jsem prováděl s různými konfiguracemi rozdílů časových značek u dat určených pro trénování i rozpoznávání, s počtem kvantilů do kterých byli klasifikovány síly signálů na přijímačích a počet k v k-NN algoritmu.

Pro minimální rozdíl časových značek dat určených pro trénování jsem zvolil hodnoty 0s, 1,5s, 3s, 5s a 8s.

Minimální rozdíl časových značek dat určených pro rozpoznávání jsem zvolil hodnoty 1,5s, 3s, 5s a 8s.

Použité kvantily pro data jsem zvolil 3 a 5 kvantilů.

Referenční algoritmus jsem nastavoval na k rovno 1, 3 a 5ti.

Následně jsem spustit učení nad daty z prvního měření z jednoho vysílače s rozpoznáváním z druhého vysílače pro tato data jsem použil kombinace všech zmíněných parametrů. Následně jsem vyměnil přijímače. Stejný postup jsem zopakoval i pro druhé měření. Použité konfigurace jsou vypsány v následující tabulce:

i	rozdíl časových značek u dat pro rozpoznávání	rozdíl časových značek u dat pro trénování	počet kvantilů HMM	počet k u k-NN
1	0	1,5	5	5
2	0	1,5	3	5
3	0	1,5	5	3
4	0	1,5	3	3
5	0	1,5	5	1
6	0	1,5	3	1
7	0	3	5	5
8	0	3	3	5
9	0	3	5	3
10	0	3	3	3
11	0	3	5	1
12	0	3	3	1
13	0	5	5	5
14	0	5	3	5
15	0	5	5	3
16	0	5	3	3
17	0	5	5	1
18	0	5	3	1
19	0	8	5	5
20	0	8	3	5
21	0	8	5	3
22	0	8	3	3
23	0	8	5	1
24	0	8	3	1
25	1,5	1,5	5	5
26	1,5	1,5	3	5
27	1,5	1,5	5	3
28	1,5	1,5	3	3
29	1,5	1,5	5	1
30	1,5	1,5	3	1
31	1,5	3	5	5
32	1,5	3	3	5
33	1,5	3	5	3
34	1,5	3	3	3
35	1,5	3	5	1
36	1,5	3	3	1
37	1,5	5	5	5
38	1,5	5	3	5
39	1,5	5	5	3
40	1,5	5	3	3
41	1,5	5	5	1
42	1,5	5	3	1

i	rozdíl časových značek u dat pro rozpoznávání	rozdíl časových značek u dat pro trénování	počet kvantilů HMM	počet k u k-NN
43	1,5	8	5	5
44	1,5	8	3	5
45	1,5	8	5	3
46	1,5	8	3	3
47	1,5	8	5	1
48	1,5	8	3	1
49	3	1,5	5	5
50	3	1,5	3	5
51	3	1,5	5	3
52	3	1,5	3	3
53	3	1,5	5	1
54	3	1,5	3	1
55	3	3	5	5
56	3	3	3	5
57	3	3	5	3
58	3	3	3	3
59	3	3	5	1
60	3	3	3	1
61	3	5	5	5
62	3	5	3	5
63	3	5	5	3
64	3	5	3	3
65	3	5	5	1
66	3	5	3	1
67	3	8	5	5
68	3	8	3	5
69	3	8	5	3
70	3	8	3	3
71	3	8	5	1
72	3	8	3	1
73	5	1,5	5	5
74	5	1,5	3	5
75	5	1,5	5	3
76	5	1,5	3	3
77	5	1,5	5	1
78	5	1,5	3	1
79	5	3	5	5
80	5	3	3	5
81	5	3	5	3
82	5	3	3	3
83	5	3	5	1
84	5	3	3	1

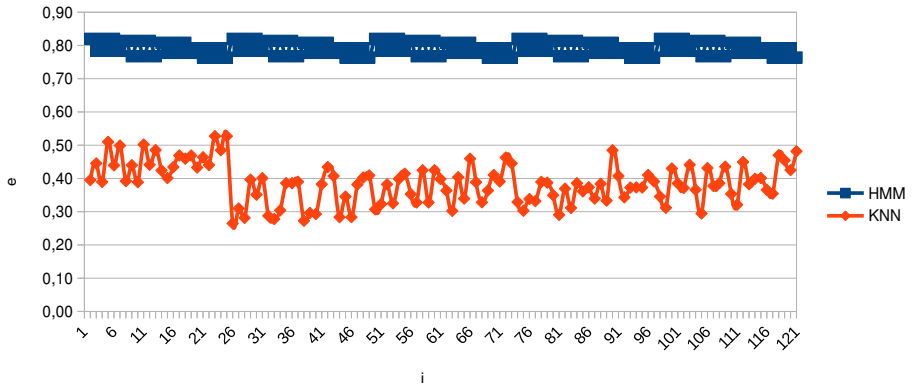
i	rozdíl časových značek u dat pro rozpoznávání	rozdíl časových značek u dat pro trénování	počet kvantilů HMM	počet k u k-NN
85	5	5	5	5
86	5	5	3	5
87	5	5	5	3
88	5	5	3	3
89	5	5	5	1
90	5	5	3	1
91	5	8	5	5
92	5	8	3	5
93	5	8	5	3
94	5	8	3	3
95	5	8	5	1
96	5	8	3	1
97	8	1,5	5	5
98	8	1,5	3	5
99	8	1,5	5	3
100	8	1,5	3	3
101	8	1,5	5	1
102	8	1,5	3	1
103	8	3	5	5
104	8	3	3	5
105	8	3	5	3
106	8	3	3	3
107	8	3	5	1
108	8	3	3	1
109	8	5	5	5
110	8	5	3	5
111	8	5	5	3
112	8	5	3	3
113	8	5	5	1
114	8	5	3	1
115	8	8	5	5
116	8	8	3	5
117	8	8	5	3
118	8	8	3	3
119	8	8	5	1
120	8	8	3	1

Tyto výpočty byli poměrně časově náročné, jejich doba se blížila k 16h na procesoru i5-3317U.

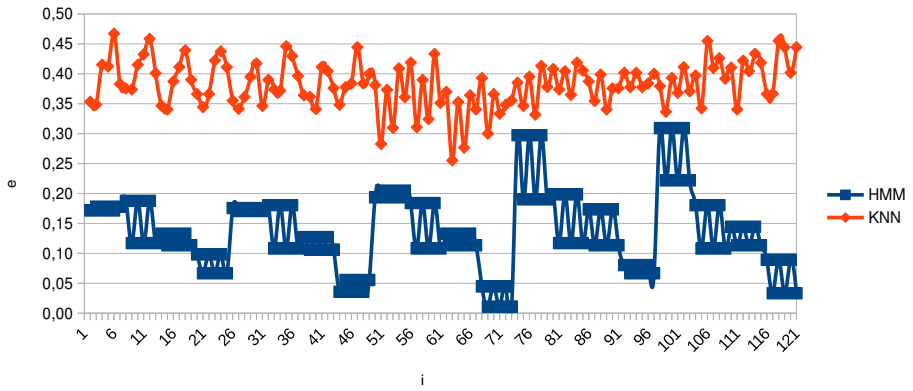
Kapitola 6
Výsledky

Každé měření a roli vysílače (jedna role pro trénování a druhá pro rozpoznávání) jsem vynesl do grafu, který znázorňuje srovnání pravděpodobností chybných rozpoznání (v grafu jako e) mezi výstupem referenčního algoritmu (k-NN) popsaného v sekci 4.3 a HMM ze sekce 4.2.5. Srovnání jsem vynesl do grafu pro každou konfiguraci (číslo konfigurace i v tabulce odpovídá ose grafu i) z tabulky vypsané v sekci 5.2.

Srovnání z prvního měření (tj. pouze jedno patro, 11500 záznamů) s daty z vysílače s označením 0x4e použitými pro trénování a z vysílače 0x4a pro rozpoznávání, lze pozorovat na grafu 12, výsledky po vyměně rolí vysílačů pak na grafu 13.



Obrázek 12: Porovnání 1. měření, učení nad 0x4e, rozpoznávání nad 0x4a

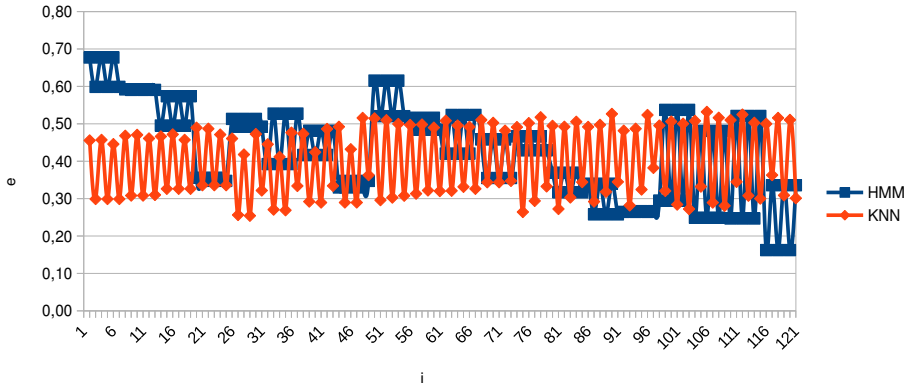


Obrázek 13: Porovnání 1. měření, učení nad 0x4a, rozpoznávání nad 0x4e

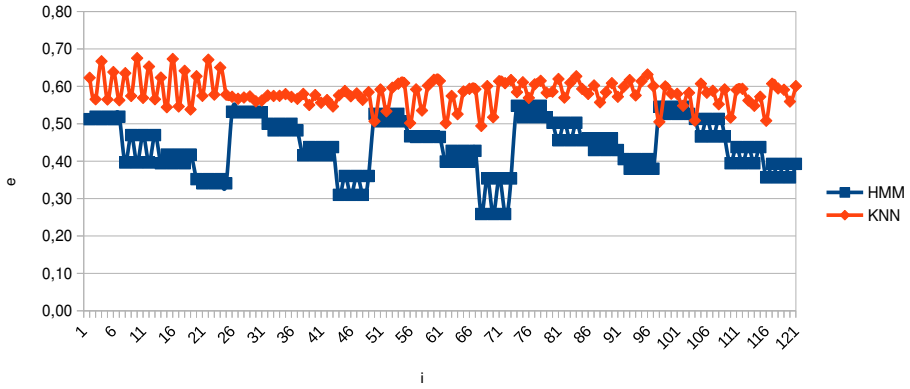
U grafu 12 si nejsem jist z jakého důvodu je metoda HMM ve všech konfiguracích

tak špatná (tj. průměrně přibližně 80% neúspěšnost), předpokládám, že to může být způsobeno častým pohybem vysílače na okrajích měřitelného prostoru, čehož jsme se při dalším měření vyvarovali.

Myslím si proto, že zajímavější jsou porovnání z druhého měření, která jsou na grafu 14 (data od vysílače 0x4e použitá pro trénování a data z vysílače 0x4a pro rozpoznávání) a na grafu 15 po vyměně rolí vysílačů.



Obrázek 14: Porovnání 2. měření, učení nad 0x4e, rozpoznávání nad 0x4a

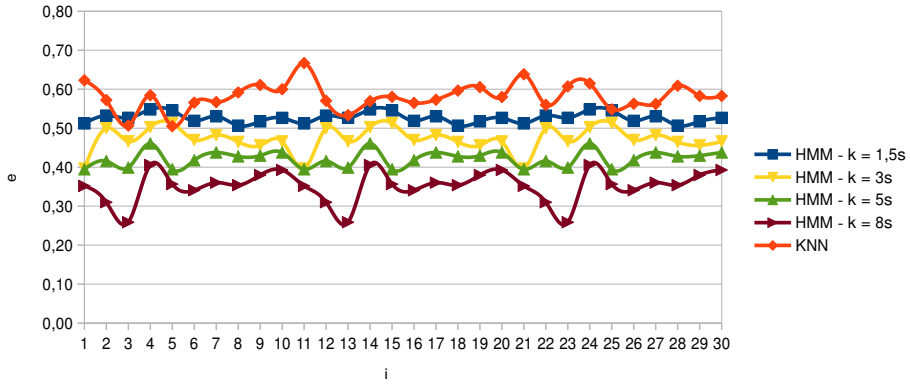


Obrázek 15: Porovnání 2. měření, učení nad 0x4a, rozpoznávání nad 0x4e

Mezi zvolenými rolemi vysílačů u prvního a druhého měření vidím jistou korelaci (i přes zmíněnou změnu subjektu provádějícího měření v sekci 5.1), proto se budu dále zaměřovat na porovnání založené na učení z vysílače 0x4a a rozpoznávání z 0x4e. Z grafů druhého měření soudím, že lépe reflektují očekávané výsledky. S přihlédnutím k pozorování kvality výsledků jednotlivých měření a jejich závislosti na zvolených rolích vysílačů jsem se dále zaměřil na detailnější porovnání výsledků vynesených do grafu 15.

V prvním detailnějším porovnání jsem seřadil výsledky podle časů minimálních odchylek mezi daty použitými pro rozpoznávání, porovnání jsem vynesl do grafu 16.

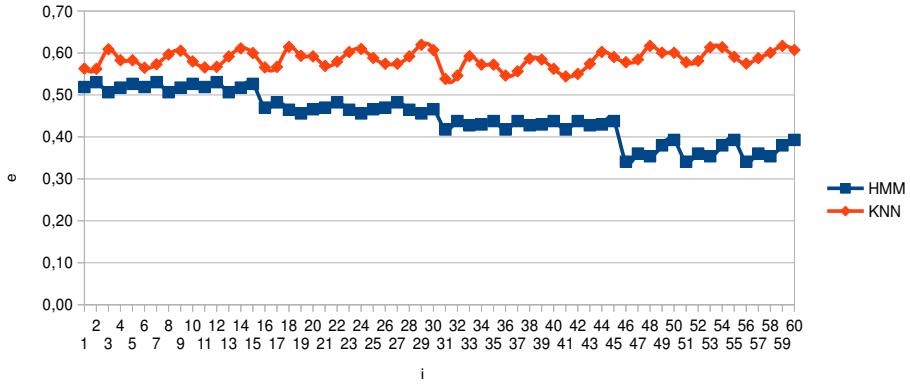
Na tomto grafu je dobře vidět ovlivnění rozdílu časových značek napříč změnou ostatních parametrů. Nejlepší výsledek pro HMM můžeme pozorovat u rozdílu k=8s u



Obrázek 16: Porovnání 2. měření, učení nad 0x4a, rozpoznávání nad 0x4e - rozděleno do skupin

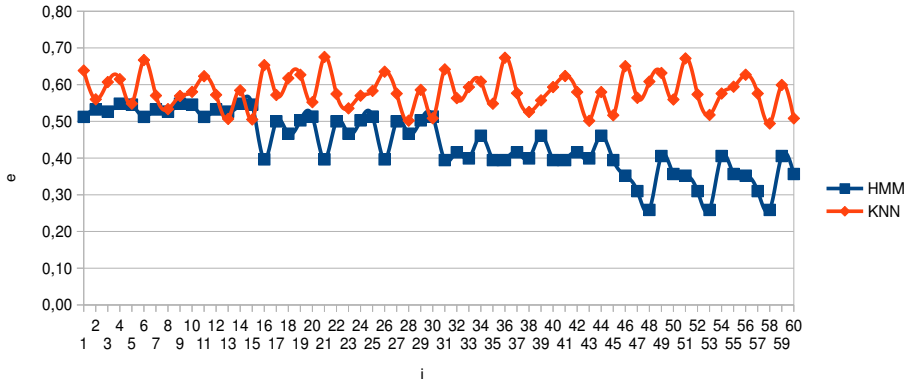
konfigurace 3, 13 a 23 (u kterých díky seřazení výsledků čísla konfigurací neodpovídají číslům konfigurací v původní tabulce, seřazená data jsou k dispozici na přiloženém CD), tyto konfigurace mají shodně nastavený rozdíl časových značek dat pro trénování a to na 3s, což v důsledku znamená, že použijeme přesnější data pro trénování a méně přesná pro rozpoznávání, jako vedlejší efekt pozitivně ovlivníme dobu výpočtu a četnost přepočítávání. Když prozkoumáme křivku k=3s můžeme si povšimnout jisté korelace s k=8s (fázově posunuté), z které vyvozuji vhodnost použití velikosti časových rozdílů u trénování a rozpoznávání v poměru přibližně 1:3.

Pro další detailnější porovnání jsem si vybral setřídění podle počtu kvantilů použitých pro HMM, porovnání jsem vynesl do grafu 17 a 18



Obrázek 17: Porovnání 2. měření, učení nad 0x4a, rozpoznávání nad 0x4e, 3 kvantily

Na těchto porovnáních mi přijde zvláštní opačná tendence, která je dobře viditelná u konfigurací (čísla konfigurací díky seřazení neodpovídají číslům konfigurací v tabulce zmíněné v sekci 5.2, seřazená data jsou k dispozici na přiloženém CD) s číslem konfigurace vetším než 46 (tj. všechny konfigurace s rozdílem časových značek větším než 8s u dat pro rozpoznávání). Porovnání založené na třech kvantilech reaguje na růst rozdílu časových značek dat pro učení negativně. Oproti tomu porovnání s pěti kvantily reaguje



Obrázek 18: Porovnání 2. měření, učení nad 0x4a, rozpoznávání nad 0x4e, 5 kvantilů

na růst rozdílu časových značek pozitivně (do dosažení zmíněné 1:3 hranice, po které se výsledky začnou prudce zhoršovat).

Kapitola 7
Závěr

Pro tuto práci jsem napsal program pro sběr dat, webovou aplikaci pro identifikaci sebraných dat a program pro trénování modelů ze sebraných dat (jak HMM, tak k-NN) a rozpoznávání nejpravděpodobnější sekvence. Oba programy jsem napsal v jazyce C++, webovou aplikaci pak v jazyce PHP.

Při práci jsem si osvojil práci s počítačovou sítí a jejími možnostmi, které jsem hojně využil při sběru dat.

Celkově jsme v průběhu práce sebrali téměř 50000 záznamů dat, nad kterými jsem prováděl trénování modelu a rozpoznávání nejpravděpodobnějších posloupností místností.

Z detailnějších porovnání výsledků ze sekce 5.2 vyvozuji jako vhodné použít 5-ti kvantilů pro úpravu dat z přijímačů, dále selekci dat s minimálním rozdílem časových značek 3s pro učení a 8s pro rozpoznávání (poměr 3:8 je velice blízko vyvozenému poměru 1:3, který se ukázal jako vhodné kritérium pro určení minimálního rozdílu časových značek). Tato konfigurace 5 kvantilů, 3s rozdíl časových značek dat pro učení a 8s rozdíl pro rozpoznávání se ukázala jako nejlepší pro HMM i referenční k-NN.

Nejlepší konfigurace HMM je neúspěšná jen na 26% naměřených datech, oproti k-NN jejíž nejlepším výsledkem je 49% neúspěšnost. Při vyhodnocování byli vyhodnoceny jako chybné rozpoznání výsledky, které označili za nejpravděpodobnější sousední místnost, což v praxi při snaze lokalizovat osobu v rozlehlých areálech nemusí být považováno jako výrazná komplikace.

Pokračováním projektu může být např. možnost upravit modelu, tak aby byl schopen využít spojitá data (v tomto projektu jsou data kvantována, díky čemu se ztrácí část naměřených informací). Další možné rozšíření je v oblasti automatizovaného opravování modelu v závislosti na napojení např. k čipovým systémům umožňujícím autorizované odemykání dveří. Možnou inovaci v uživatelské části vidím v implementování interaktivního zobrazování aktuální pozice na mapě (tj. otočení funkce webové aplikace specializované pro sběr dat).

Literatura

[1] Triangulace – wikipedie. http://cs.wikipedia.org/wiki/Triangulace. Naposledy navštíveno 19. 5. 2015.

[2] M. Ibrahim; Y. Moustafa. A hidden markov model for localization using low-end gsm cell phones. http://arxiv.org/pdf/1010.3411.pdf.

[3] Libor Urbaník. Methods for people localization based on radio signals.

[4] Tomáš Procházka Michal Houdek, Tomáš Svoboda. Klasifikace podle nejbližších sousedů nearest neighbour classification [k-nn]. http://cmp.felk.cvut.cz/cmp/courses/recognition/zapis_prednasky/zapis_01/4/rpz4.pdf. Naposledy navštíveno 12. 5. 2015.

[5] R. Arthi; K. Murugan. Localization in wireless sensor networks by hidden markov model.

[6] CSc. Prof. Ing. Jan Nouza. Pokročilé metody rozpoznávání řeči. http://itakura.ite.tul.cz/jan/PMR/prednaska4.pdf.

[7] Stephen Tu. Derivation of baum-welch algorithm for hidden markov models. http://www.cs.berkeley.edu/~stephentu/writeups/hmm-baum-welch-derivation.pdf. Naposledy navštíveno 11. 2. 2015.

[8] Václav Hlaváč. Rozpoznávání s markovskými modely. http://cmp.felk.cvut.cz/~hlavac/TeachPresCz/31Rozp/61MarkovianPR.pdf. Naposledy navštíveno 11. 5. 2015.

[9] VOCAL Technologies. Viterbi algorithm in speech enhancement and hmm. http://www.vocal.com/echo-cancellation/viterbi-algorithm-in-speech-enhancement-and-hmm/. Naposledy navštíveno 18. 5. 2015.

[10] Oracle Corporation. Mysql. http://www.mysql.com/. Naposledy navštíveno 8. 2. 2015.

[11] MariaDB Corporation. Mariadb. https://mariadb.com/. Naposledy navštíveno 8. 2. 2015.

[12] Oracle Corporation. Mysql :: Mysql 5.0 reference manual :: 14.2 the innodb storage engine. http://dev.mysql.com/doc/refman/5.0/en/innodb-storage-engine.html. Naposledy navštíveno 8. 2. 2015.

[13] Oracle Corporation. Mysql :: Download connector/c++. http://dev.mysql.com/downloads/connector/cpp/. Naposledy navštíveno 8. 2. 2015.

[14] Oracle Corporation. Mysql :: Mysql connector/c++ developer guide :: 7.6 mysql connector/c++ complete example 2. http://dev.mysql.com/doc/connector-cpp/en/connector-cpp-examples-complete-example-2.html. Naposledy navštíveno 10. 2. 2015.

[15] Nette Foundation. Nette framework. http://nette.org/. Naposledy navštíveno 10. 2. 2015.

[16] Jan Trejbal. Trejjam/authorization. https://github.com/Trejjam/Authorization. Naposledy navštíveno 10. 2. 2015.

[17] Filip Procházka. Kdyby/console. https://github.com/kdyby/console. Naposledy navštíveno 10. 2. 2015.

[18] MathWorks. Matlab engine api for c, c++, and fortran - matlab & simulink. http://www.mathworks.com/help/matlab/calling-matlab-engine-from-c-c-and-fortran-programs.html. Naposledy navštíveno 11. 5. 2015.

[19] MathWorks. Hidden markov model parameter estimates from emissions and states - matlab hmmestimate. http://www.mathworks.com/help/stats/hmmestimate.html. Naposledy navštíveno 7. 5. 2015.

[20] MathWorks. Hidden markov model most probable state path - matlab hmmviterbi. http://www.mathworks.com/help/stats/hmmviterbi.html. Naposledy navštíveno 11. 5. 2015.

Obsah CD

- Tato bakalářská práce
- Program pro sběr dat
- Program pro provádění výpočtů
- Webová aplikace umožňující sběr dat
- Naměřená data
- Konfigurace použité v části "Výsledky"