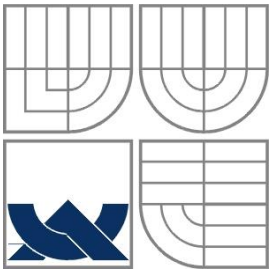
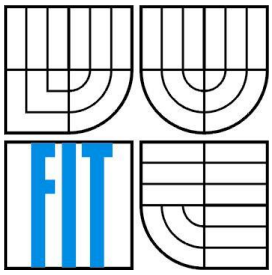


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NÁVRH SYSTÉMU PRO PRECIZNÍ LOKALIZAČNÍ SLUŽBY

DESIGN OF A SYSTEM FOR PRECISE LOCALIZATION SERVICES

DIPLOMOVÁ PRÁCE

MASTER‘S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Martin Krippel

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Vladimír Veselý

BRNO 2016

Zadání diplomové práce/18990/2015/xkripp00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2015/2016

Zadání diplomové práce

Řešitel: **Krippel Martin, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Návrh systému pro precizní lokalizační služby**
Design of a System for Precise Localization Services
Kategorie: Počítačové sítě

Pokyny:

1. Analýza stávajících lokalizačních systémů pro interiérové aplikace na báze TDOA s využitím multilaterace a algoritmů v nich využívaných.
2. Návrh a implementace filtru pro nízko-úrovňová lokalizační data.
3. Návrh a implementace predikčního filtru pro finální vyhlazení a stabilitu pohybu.
4. Vytvoření grafického simulačního nástroje pro posouzení instalace a rozložení jednotek lokalizačního systému.
5. Otestování a porovnání navrženého systému a implementovaných algoritmů. Diskuze o činnosti při nasazení v reálném prostředí.

Literatura:

- P. Kim, Kalman Filter for Beginners, CreateSpace Independent Publishing Platform, ISBN: 978-1463648350
- A. Malik, RTLS For Dummies, Wiley, ISBN: 978-0-470-39868-5, April 2009

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2 včetně.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Veselý Vladimír, Ing.**, UIFS FIT VUT
Konzultant: Šimek Milan, Ing., FEKT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 25. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Cieľom tejto diplomovej práce bolo analyzovať bezdrôtovú lokalizáciu v interiéri. Analyzované sú niektoré technológie bezdrôtovej lokalizácie ako Time of Arrival alebo Time Difference of Arrival. V práci je taktiež popísaný systém spoločnosti SEWIO. Hlavnou časťou je popis, návrh a implementácia Kalmanovho filtra. Kalmanov filter je použitý na vylepšenie dvojrozmerných pozičných dát a pri synchronizácii kotiev (zariadenia na určenie polohy objektu v systéme SEWIO). Je popísaných niekoľko systémových modelov pre Kalmanov filter.

Abstract

The aim of this term project was to analyze wireless indoor localization. It contains analysis of some wireless localization techniques such as Time of Arrival or Time Difference of Arrival. The paper also describes the system of SEWIO Company. Main part of the master's thesis is description, design and implementation of the Kalman filter. The Kalman filter is used to improve two-dimensional positional data and synchronization of anchors (devices for finding a position of an object in SEWIO system). There are described a few system models for the Kalman filter.

Kľúčové slová

Bezdrôtová interiérová lokalizácia, TOA, TDOA, Kalmanov filter, lokalizačné techniky.

Keywords

Indoor wireless localization, TOA, TDOA, Kalman filter, localization techniques.

Citácia

Krippel Martin: Návrh systému pro precizní lokalizační služby, diplomová práce, Brno, FIT VUT v Brně, 2016

Návrh systému pro precizní lokalizační služby

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením Ing. Vladimíra Veselého.
Ďalšie informácie mi poskytol Ing. Tomáš Kočan.
Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

……
Martin Krippel
24.5.2016

Poďakovanie

Chcel by som sa poďakovať Ing. Vladimírovi Veselému za vedenie práce a Ing. Tomášovi Kočanovi za všetky potrebné informácie, ktoré som potreboval, veľkú ochotu a trpezlivosť.

Za ochotu a najmä opravu chýb v mojej práci by som rád venoval Ing. Vladimírovi Veselému moje najobľúbenejšie (resp. najčastejšie pripravované) jedlo. Sú to cestoviny s vajcom. Príprava je veľmi jednoduchá. Na jednu porciu sú potrebné: cestoviny (150 g), cibuľa (1-2 ks), olej a samozrejme vajcia (2-3 ks). Dajú sa variť cestoviny a pripraví sa panvica s troškou oleja. Popri varení cestovín sa očistí a nakrája cibuľa na drobné kúsky. Nakrájaná cibuľa sa usmaží do zlatista panvici. Keď sú cestoviny hotové scedia sa a pridajú na panvicu k osmaženej cibuľke. Cestoviny a cibuľka sa premieša a pridajú sa rozšľahané vajcia. Všetko dohromady sa dobre premieša a smaží sa to, kým nie sú vajíčka hotové (majú konzistenciu ako pri praženici). Cestoviny s vajcom je možné dochutiť kečupom, sladkou paprikou či mletým korení. Pre každého podľa jeho chuťových preferencií.

Obsah

1 Úvod

Lokalizácia sa stala bežnou súčasťou nášho života. Najznámejšia je zrejme GPS, používaná v smartfónoch a navigáciách [1]. V takýchto systémoch sa ako zdroje signálu používajú napr. satelity. Problémom takýchto systémov je, že sa vôbec nedajú použiť vo vnútri budov. Signály zo satelitov sú oslabené, rozptýlené vďaka rôznym prekážkam ako sú steny či strechy. Zároveň v niektorých prípadoch by mohla byť chyba v GPS lokalizácii v nejakej budove aj niekoľko metrov a lokalizovaný objekt sa razom ocitol mimo budovy alebo vo vedľajšej miestnosti. Podľa oficiálnej stránky americkej vlády poskytujúcej informácie o GPS [2], presnosť, ktorú môže užívateľ dosiahnuť, je závislá na týchto faktoroch:

- Atmosférické efekty (vlhkosť, atmosférický tlak, a i.);
- Zastrený výhľad na oblohu (*angl. sky blockage*) – termín, ktorý popisuje stav, kedy nie je priamy výhľad na oblohu z miesta GPS prijímača. Výhľad môže byť narušený vysokými budovami, stromami, a i.;
- Kvalita prijímača.

Reálne dáta od FAA[1] (Federal Aviation Administration – Federálna letecká správa) ukazujú, že ich vysoko kvalitné prijímače dosahovali horizontálnu presnosť lepšiu ako 3,5 metra. Dané údaje platia pre civilnú GPS.

Aj na základe týchto argumentov je zrejmé, že GPS signál nie je vhodný pre všetky situácie. GPS je bežne nepoužiteľné pre miesta ako tunely či budovy. A preto je vhodné, dokonca žiaduce, používať iný systém pre lokalizáciu v interiérových priestoroch.

V tejto práci venovanej interiérovej lokalizácií sa preto budem najprv venovať najprv systémom a technikám používaným v danom probléme. Ďalej budem rozoberať filtrovanie dát pre presnejšie určenie polohy a Kalmanov filter pre vylepšenie pozičných dát.

V kapitole 2 sú popísané najpoužívanejšie techniky bezdrôtovej lokalizácie objektov. Zameral som sa hlavne na techniku Time Difference of Arrival (TDOA), ktorú používa systém SEWIO. V kapitole 3 je popísaný interiérový lokalizačný systém spoločnosti SEWIO. Popísaná je architektúra ich systému a proces lokalizácie. V kapitole 4 sa nachádza popis Kalmanovho filtra a jeho implementácia. Táto kapitola obsahuje popis niekoľkých systémových modelov, pre ktoré je možné použiť Kalmanov filter. V kapitole 5 je popísaná implementačná časť diplomovej práce. V závere je moje zhodnotenie tejto práce.

[1] http://www.gps.gov/systems/gps/performance/accuracy/

2 Analýza interiérovej bezdrôtovej lokalizácie

Interiérový lokalizačný systém (*angl. indoor positioning system - IPS*) je systém na určenie polohy objektov alebo ľudí vo vnútri budov použitím rádiových vĺn. IPS používa rôzne techniky na určenie polohy hľadaného objektu. Najznámejšou technikou je triangulácia.

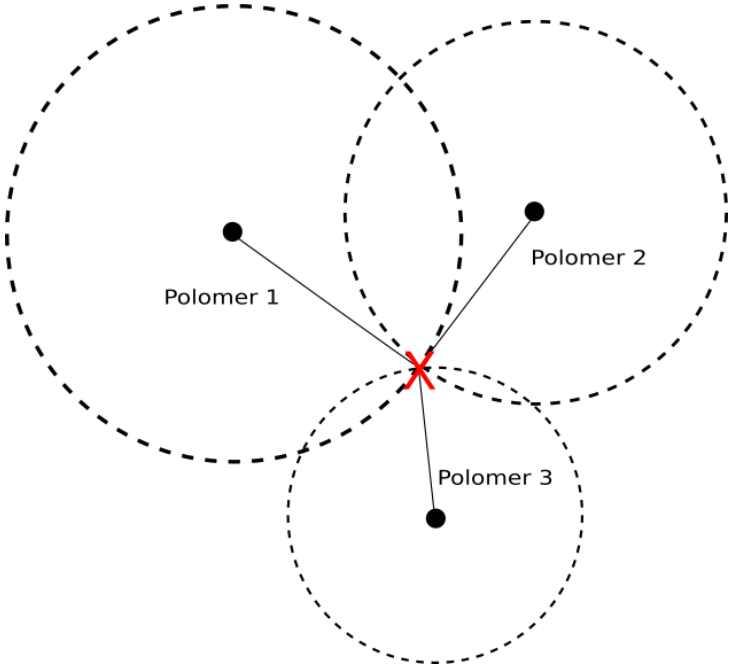
Triangulácia je lokalizačná technika využívajúca vlastnosti trojuholníkov k vypočítaniu pozície objektu. Je možné rozdeliť ju na dve časti v závislosti na použitom spôsobe výpočtu pozície hľadaného objektu. Sú to laterácia, kedy sa pre výpočet pozície použije nameraná vzdialenosť a angulácia, používa sa meranie pomocou uhlov. Pri písaní nasledujúcej časti o laterácií a angulácií som vychádzal z materiálov [3], [4] a [16].

2.1 Laterácia

Laterácia (niekedy označovaná aj ako trilaterácia alebo multilaterácia) je proces, kedy sa na základe nameraných vzdialeností z referenčných bodov určí pozícia. Počítanie polohy objektov v 2D priestore vyžaduje aspoň tri ne-kolineárne referenčné body. Princíp je ukázaný na obrázku 2.1.

Na obrázku 2.1 sú tri prijímače (označené čiernymi bodmi umiestnené na známych pozíciách) a vysielač (označený červeným písmenom X). Vysielač je od každého prijímača vzdialený o určitú vzdialenosť naznačenú pomocou úsečky s popisom „*Polomer*“. Výsledná poloha vysielača sa spočíta ako prienik jednotlivých kružníc so stredmi na pozíciách prijímačov a polomerom určeným vzdialenosťou vysielača od daného prijímača.

V trojdimenzionálnom priestore sú potrebné aspoň štyri ne-koplanárne referenčné body.



Obr. 2.1: Princíp zistenia pozície v 2D priestore pomocou laterácie

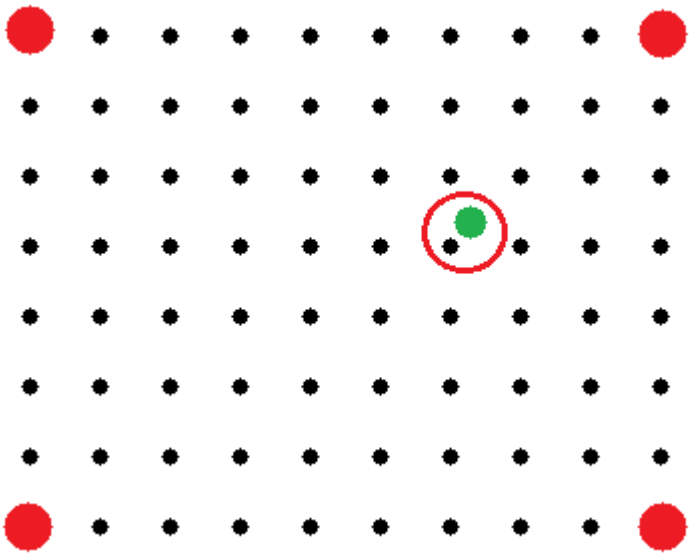
2.1.1 TOA – Time of Arrival

Jednou z techník na výpočet vzdialenosti hľadaného objektu (vysielača) od referenčných bodov (prijímačov) je TOA – Time of Arrival (dá sa preložiť ako čas príchodu). Pre výpočet v dvojrozmernom priestore musí byť meranie TOA aspoň z troch referenčných bodov. Vzdialenosť z hľadaného bodu sa prepočíta na základe času, ktorý bol potrebný na prijatie signálu z vysielača prijímačom. Vo všeobecnosti má takýto priamy výpočet TOA dva problémy. Prvým je synchronizácia. Vysielače aj prijímače musia byť veľmi precízne zosynchronizované. Druhým je fakt, že v prenášanom signáli musí byť zaznamenaná časová značka. Na základe tejto hodnoty je TOA schopné vypočítať vzdialenosť medzi vysielačom a prijímačom.

Poloha objektu sa vypočíta priamočiaro ako prienik kružníc, ktoré majú stred v referenčných bodoch. Vzhľadom k nepresnostiam (spôsobenými rušením signálu počas prenosu) sa kružnice nepretnú v jednom bode, ale vznikne malá oblasť. V tejto oblasti sa pozícia určí napríklad pomocou least-square algoritmu (algoritmus najmenších štvorcov). Iné algoritmy môžu byť closest-neighbor (algoritmus najbližšieho suseda) alebo residual weighting.

Closest-Neighbor

Algoritmus Closest-Neighbor (pre TOA sa používa CN-TOAG, kde písmeno „G“ znamená mriežka – *angl. grid*) určuje polohu na základe pravidelnej mriežky referenčných bodov. Body sú rozmiestnené medzi vysielačmi. Majú predpočítanú hodnotu vzdialeností od každého vysielača. Takže každý bod obsahuje vektor vzdialeností od vysielačov. Na základe vzdialeností objektu od vysielačov sa jeho poloha určí ako poloha najbližšieho bodu referenčnej mriežky. Princíp je dobre vidieť na obr. 2.2. Červenými bodmi sú znázornené vysielače, zeleným lokalizovaný objekt a čierne body sú referenčnou mriežkou. Červenou kružnicou je zobrazený výber polohy v priestore ako poloha „najbližšieho suseda“.



Obr. 2.2: Princíp CN algoritmu

Least-square algoritmus

Least-square algoritmus (LS) sa v podstate zaoberá minimalizovaním hodnoty funkcie, ktorá má zvyčajne tvar:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \left(\sqrt{(x - x_i)^2 + (y - y_i)^2} - d_i \right)^2$$

kde N je počet vysielačov, x_i a y_i sú body určujúce polohu vysielač, body x a y určujú nameranú polohu v Karteziánskom súradnicovom systéme a d_i je vzdialenosť nameranej polohy od daného vysielača. Rozdiel v zátvorkách sa nazýva reziduál (*angl. residual*). V ideálnom prípade, keby určenie polohy prebehlo bez chyby, by hodnota v zátvorkách bola nulová a teda aj výsledná funkcia $f(\boldsymbol{x})$ by sa rovnala 0. Avšak meranie vzdialenosti obsahuje nejaké chyby, a preto nebude hodnota funkcie presne 0. Riešenie, body x a y, sa môže určiť minimalizovaním hodnoty $f(\boldsymbol{x})$.

Existuje viacero implementácií LS algoritmu. Tieto implementácie sa líšia postupov v hľadaní minimálnej hodnoty $f(\boldsymbol{x})$. Príkladom môžu byť Davidonov algoritmus[16] alebo Levenberg-Marquardtov algoritmus[17].

Residual Weighting

Tento algoritmus vyžaduje väčší počet vysielačov, ako je minimálny potrebný počet (teda pre TOA viac ako tri). Takže ak je počet vzdialenostných meraní väčší ako požadované minimum, vzdialenostné merania môžu byť rozdelené do niekoľkých rôzny podskupín (vysielačov). Z týchto podskupín sa odvodia priebežné odhady reziduálov pomocou LS algoritmu. Konečný odhad pozície môže byť vypočítaný ako lineárna kombinácia priebežných odhadov, pričom každý odhad je váhovaný obrátenou hodnotou daného reziduálu. To znamená, že vo výpočte budú mať reziduály s nižšou hodnotou väčšiu váhu. Týmto spôsobom sa zvyšuje presnosť odhadu.

Ak máme M meraní vzdialenosti ($M > 3$), algoritmus vytvorí N rôznych kombinácií vzdialenostných meraní:

$$N = \sum_{i=3}^{M} \binom{M}{i}$$

kde každá kombinácia je reprezentovaná sadou $\{S_k \mid k = 1, 2, \ldots, N\}$. Pre každú sadu meraní sa spočíta odhad pomocou LS algoritmu. Keďže sada S_k nemusí mať pre všetky kombinácie rovnakú veľkosť a odhad môže závisieť ak od veľkosti sady, pre odstránenie tejto závislosti sa počíta pre každý priebežný odhad ($\boldsymbol{x'_k}$) normalizovaný reziduál:

$$\tilde{R}_{es}(\boldsymbol{x'_k}, S_k) = \frac{R_{es}(\boldsymbol{x'_k}, S_k)}{sizeOf\ S_k}$$

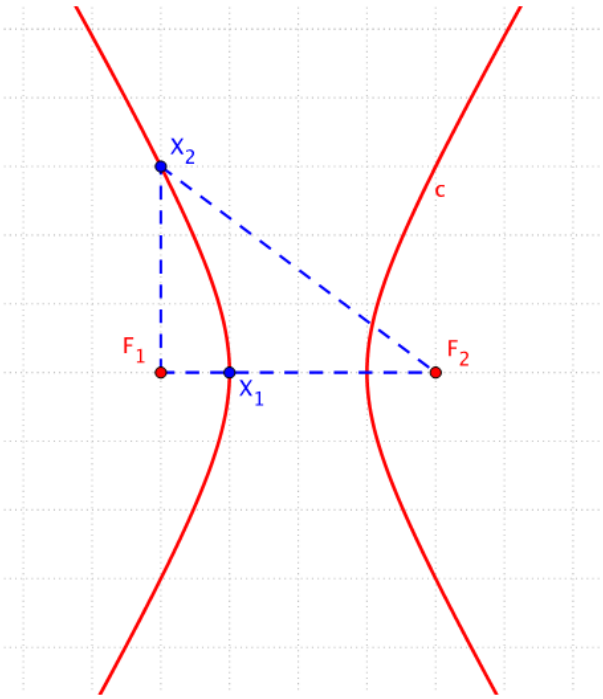
Výsledný odhad polohy, $\boldsymbol{x'}$, sa spočíta:

$$\boldsymbol{x'} = \frac{\sum_{k=1}^{N} \boldsymbol{x'_k} \left(\tilde{R}_{es}(\boldsymbol{x'_k}, S_k) \right)^{-1}}{\sum_{k=1}^{N} \left(\tilde{R}_{es}(\boldsymbol{x'_k}, S_k) \right)^{-1}}$$

2.1.2 TDOA – Time Difference of Arrival

Ďalším spôsobom určenia polohy vysielača je TDOA – Time Difference of Arrival. Pre svoj výpočet používa časový rozdiel príchodu signálu na viaceré (aspoň dva pre jeden výpočet TDOA) prijímače. Netreba teda poznať vzdialenosť od vysielača k prijímaču, ako pri TOA. Vysielač leží na hyperboloide (v dvojrozmernom priestore je to hyperbola) získanom z časového rozdielu medzi dvoma prijímačmi.

Hyperbola je množina všetkých bodov v danej rovine, ktoré majú konštantnú absolútnu hodnotu rozdielu vzdialeností od dvojice pevne daných bodov [11]. Príklad hyperboly je zobrazený na obr. 2.3.



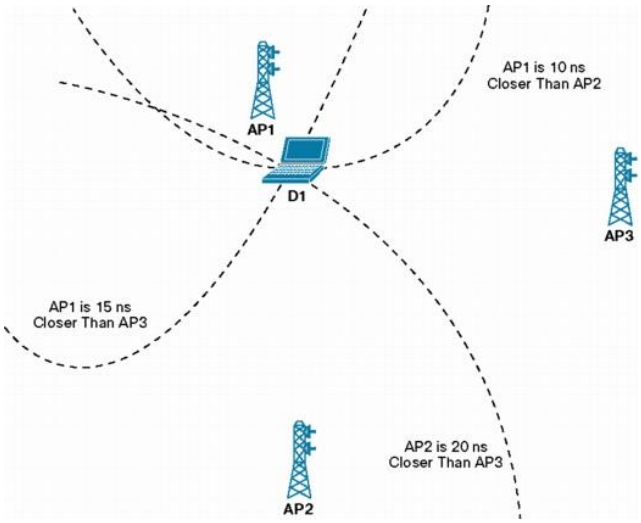
Obr. 2.3: Hyperbola[11]

Keďže prijímače majú pevne stanovenú polohu a je známy časový rozdiel medzi jednotlivými dvojicami, na základe danej definície je zrejmé, prečo vysielač leží na prieniku hyperbol. Hyperboloid získame rotáciou hyperboly okolo jej osi. Rovnica hyperboloidu je nasledovná:

$$R_{i,j} = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} - \sqrt{(x_j - x)^2 + (y_j - y)^2 + (z_j - z)^2}$$

kde (x_i, y_i, z_i) a (x_j, y_j, z_j) sú fixné súradnice prijímačov a (x, y, z) sú súradnice hľadaného vysielača. Podobne ako pri TOA, kde sa objekt nachádzal v prieniku kružníc, sa tentokrát objekt nachádza v prieniku hyperbol (príp. hyperboloidov v trojrozmernom priestore). Z toho vyplýva, že je znova potrebné mať aspoň tri fixne umiestnené prijímače.

Na obr. 2.4 je znázornený princíp TDOA. Popisuje jednotlivé časové rozdiely medzi prijímačmi (označenými AP1, AP2 a AP3). Znázornené hyperboly majú ohniská v mieste daných prijímačov. Hľadaný objekt (D1) sa nachádza na prieniku hyperbol.



Obr. 2.4: Princíp lokalizácie pomocou TDOA[10]

Taktiež ako pri TOA aj pri TDOA vzniká nepresnosť. Táto nepresnosť je podobná ako pri použití TOA. Podrobnejšie štúdie na porovnanie presnosti TOA a TDOA alebo porovnanie chybových charakteristík sú popísané v [5], respektíve v [6]. V prvej zo spomínaných prác sa pomocou simulácií metódou Monte Carlo porovnáva presnosť meraní pomocou TOA a TDOA. V druhej [6] sa porovnávajú chybové charakteristiky TOA a TDOA. Dôraz sa kladie na ich presnosť odhadu pozície, kovariačnú chybu a DOP (*angl. Dilutions of precision*). DOP je termín používaný v satelitnej navigácií na popis vplyvu satelitnej konfigurácie vzhľadom na presnosť dát získaných z GPS prijímačov [12].

Aj keď v oboch prácach sa porovnáva TOA a TDOA z rozličných pohľadov, záver oboch znie podobne. A to, že obe meracie techniky majú takmer rovnakú presnosť.

Výhodou TDOA oproti TOA je fakt, že vysielač a prijímač nemusia byť vzájomne zosynchronizované. Synchronizácia je potrebná len medzi prijímačmi.

2.1.3 RTOF – Roundtrip Time of Fligth

Táto metóda meria „čas letu“ signálu (*angl. Roundtrip Time of Flight*), za ktorý sa signál dostal od vysielača k prijímaču a späť. Ide o veľmi podobný princíp ako TOA. Miernejšia relatívna synchronizácia nahrádza už spomínanú synchronizáciu pri TOA, ktorá musí byť veľmi precízna. Vzdialenosť medzi vysielačom a prijímačom sa počíta z času letu signálu.

Najprv lokalizačná jednotka (hľadaný objekt) vyšle signál transpondérom (majú definovanú presnú polohu v priestore). Transpondér deteguje daný signál. Po krátkej dobe čakania, ktorá je individuálne preň nastavená, každý transpondér pošle zosilnenú invertovanú kópiu signálu späť lokalizačnej jednotke. Vzájomnou krížovou koreláciou (*angl. cross-correlation*) poslanej sekvencie s prijatými demodulovanými signálmi je vzdialenosť r_i ku každému transpodéru i vypočítaná ako rozdiel celkového času letu signálu tam a späť $t_{RTOF,i}$ a doby čakania transpondéru $t_{w,i}$ (čas, ktorý transpondér čakal na odoslanie opačného signálu). Od tejto hodnoty je ešte odčítaný kalibračný offset $r_{offset,i}$.

$$r_i = \left(t_{RTOF,i} - t_{w,i}\right) * \frac{c_0}{2} - r_{offset,i}$$

V predchádzajúcej rovnici je c_0 rýchlosť svetla. Offset $r_{offset,i}$ je zdržanie spôsobené spracovaním signálu v transpondére a lokalizačnej jednotke. Táto hodnota sa dá zistiť kalibračným

meraním pri známej vzdialenosti. Lokalizačná jednotka pošle spočítanú vzdialenosť, tri najvyššie vrcholy amplitúd a časovú známku do pripojeného PC, kde sa odhadne pozícia.

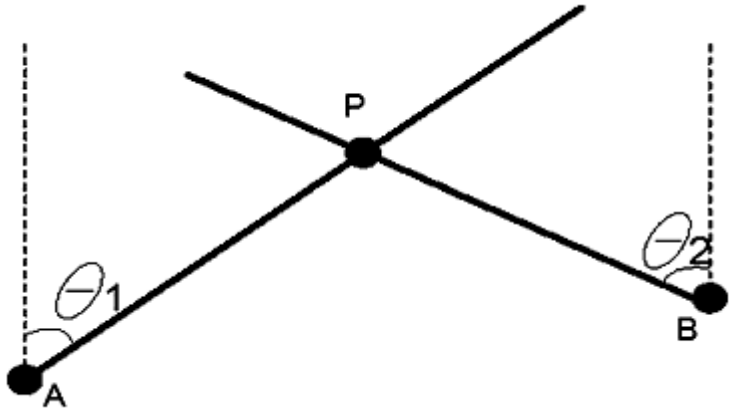
Odhad polohy sa vykonáva jednotlivo pre každý meraný cyklus. V určitom časovom kroku sú známe vzdialenosti medzi transpondérom *i* a lokalizačnou jednotkou. Každý vektor $\boldsymbol{r}_i$ obsahuje tri vzdialenosti pre každý z troch najvyšších vrcholov výslednej korelácie. Systém taktiež obsahuje vrcholy amplitúd pre dané tri vrcholy korelácie. Po aplikovaní prahu na tieto vrcholy kvôli odstráneniu šumu a falošných vrcholov sa z troch vzdialeností v $\boldsymbol{r}_i$ vyberie tá najmenšia. Týmto spôsobom sa pre každý transpondér získa vzdialenosť k lokalizačnej jednotke. Poloha lokalizačnej jednotky sa určí ako prienik kružníc so stredmi v pozíciách transpondérov. Polomer je daná vzdialenosť od lokalizačnej jednotky [13].

2.2 Angulácia

Angulácia (známa tiež ako triangulácia) používa na určenie polohy hľadaného objektu uhly. Objekt je možné nájsť v prieniku niekoľkých polpriamok, ktorých smer je určený pomocou zmerania uhlov. Táto metóda potrebuje minimálne dva referenčné body so známymi súradnicami a dva zmerané uhly pre určenie polohy objektu v dvojrozmernom priestore.

Ako je vidno na obr. 2.5 pri angulácií stačia dva referenčné body so známymi súradnicami pre určenie polohy v dvojrozmernom priestore. Ekvivalentne v trojrozmernom priestore stačia tri referenčné body. To je oproti laterácií výhoda, keďže vo všeobecnosti pri laterácií treba aspoň o jeden referenčný bod viac, ako je počet dimenzií v danom priestore. Nevýhodou sú relatívne vysoké a komplexné hardvérové požiadavky a pokles presnosti lokalizácie, keď sa objekt vzďaľuje od meracích jednotiek.

Na obr. 2.5 sú zobrazené dva body A a B so známou pozíciou. V každom z nich sa nachádza smerová anténa. Pohyblivý objekt je označený písmenom P. Smerové antény zachytia signál vyslaný pohyblivým objektom. Na základe smeru, z ktorého signál dorazil, zostroja myslené polpriamky vychádzajúce z referenčných bodov (A, B). V prieniku vytvorených sa nachádza hľadaný objekt (P).



Obr. 2.5: Princíp angulácie [4]

3 Analýza lokalizačného systému SEWIO

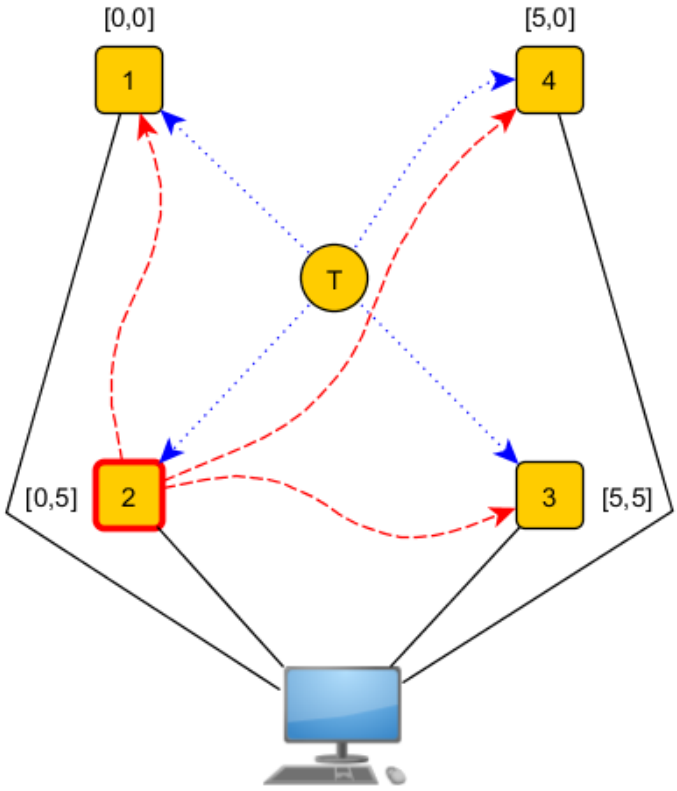
Lokalizačný systém je zložený z niekoľkých častí. Najdôležitejšími časťami sú tagy a kotvy. Zloženie celého systému sa dá popísať nasledovne [7]:

- Tag – je malé (ideálne čo najmenšie) bezdrôtové zariadenie s lokalizačnou technológiou. Je využívaný na lokalizáciu objektov alebo ľudí. Tagy môžu mať rôznu formu. Či už sú pripevnené k predmetom (ako súčasť hľadaných objektov) alebo sú nosené (náramky, opasky, a i.);
- Kotva (polohové zariadenie) – je zariadenie so známou nemennou polohou. Pomocou kotiev sú lokalizované tagy v priestore. Lokalizácia prebieha na základe merania pseudovzdialenosti (termín pseudovzdialenosť vyjadruje fakt, že vzdialenosť medzi tagom a kotvou je určená časom letu signálu medzi zariadeniami a rýchlosťou signálu) medzi tagom a kotvou. Využíva sa princíp TDOA popísaný v predchádzajúcej kapitole;
- Jadro lokalizácie – softvér komunikujúci s kotvami a tagmi, ktorý na základe získaných informácií vypočíta polohu tagu;
- Middleware – softvér, ktorý spája lokalizačný systém (tagy, kotvy a jadro) s aplikáciou;
- Aplikácia – front-end zobrazujúci spracované informácie (webová stránka).

Kotvy sú rozmiestnené v priestore (hala, kancelária, a i.) na známej pozícii. Všetky kotvy sú pripojená k serveru, na ktorom sa počíta poloha tagu. Počet kotiev záleží na rozmeroch priestoru, ale aj na požadovanej presnosti. Kotvy musia byť vzájomne časovo zosynchronizované (synchronizácia je popísaná v ďalšej časti tejto kapitoly). Minimálne jedna kotva musí byť tzv. master, podľa ktorej sa zosynchronizujú časy na kotvách. Tag (tagy) sa pohybuje v priestore vytýčenom kotvami a v pravidelných intervaloch im posiela správy. Príklad ako môže vyzerať lokalizácia je na obr. 3.1.

Na obrázku sú znázornené:

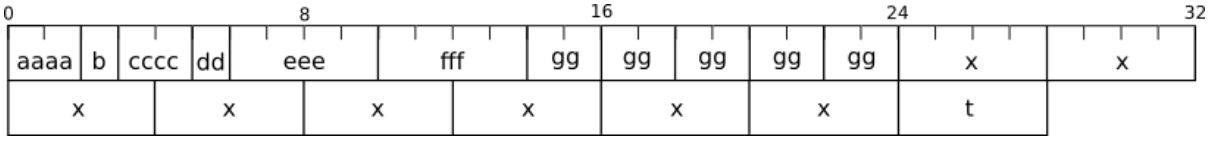
- Kotvy – žlté štvorce označené číslami 1-4;
  Červene ohraničená kotva (2) je master. Každá kotva má absolútne súradnice, ktoré ju identifikujú v priestore;
- Tag – žltý kruh označený písmenom T;
- Server – zobrazený ako desktop;
- Prepojenie kotiev a desktopu – čierne čiary;
- Synchronizačný signál pre synchronizáciu kotiev – červené šípky;
- Správa tagu poslaná na všetky kotvy – modré šípky.



Obr. 3.1: Jednoduchý príklad lokalizačného systému

3.1 Synchronizácia časov na kotvách

V systéme SEWIO posielajú kotvy na server nasledujúce správy o svojej činnosti. Formát správy je zobrazený na obr. 3.2.



Obr. 3.2: Formát správy

- aaaa – hexadecimálne číslo označujúce kotvy, ktoré prijali signál z kotvy alebo tagu. Označenie kotiev je v rozsahu 0000 – FFFF, kde 0 na prvom mieste definuje, že sa jedná o kotvu;
- b – označenie typu správy. ASCII znak „E“ označuje synchronizačnú správu. V opačnom prípade je dané pole vyplnené ASCII znakom „A“;
- cccc – hexadecimálne číslo označujúce zariadenie, ktoré správu poslalo. Označenie je v rozsahu x000 – xFFF, kde „x“ definuje typ zariadenia (0 – kotva, 1 – tag);
- dd – označenie typu správy. Kód „bb“ definuje blink správu (správa poslaná kotvou reagujúca na signál vyslaný tagom). V opačnom prípade je dané pole vyplnené kódom „aa“;
- eee – skupinové sekvenčné číslo;
- fff – sekvenčné číslo v rámci skupiny „eee“;

- gg – hexadecimálne číslo. Pätica čísel „gg“ definuje hodnotu čítača na kotve, ktorá bola aktuálna v čase prijatia správy. Z tejto hodnoty sa následne určí čas na základe nasledujúceho vzorca. Pätica čísel „gg“ sa prevedie do decimálnej hodnoty. Táto decimálna hodnota sa predelí konštantou[2] (čas navýšenia čítača o hodnotu jedna v sekundách). Výsledkom je čas v sekundách;
- x – číslo typu `float` alebo `integer`. Na základe tejto 8-tice čísel „x“ sa dá určiť sila signálu (RSSI);
- t – teplota MCU na kotve, ktorá odosiela správu. Hodnota „t“ je typu `float`. Teplota je udávaná v °C.

Príklady reálnych správ s popisom (znak „|“ v správach má funkciu oddeľovača):

- synchronizačná správa
  |0001|E|0001|aa|034|136|3b|3e|87|fe|55|57.63|
  Kotva 0001 (master) vyslala žiadosť o synchronizáciu. Správa má sekvenčné číslo 136 a patrí do skupiny označenej sekvenčným číslom 034. Žiadosť bola poslaná v čase definovanom hodnotou čítača 3b3e87fe55;
- blink správa
  |0006|A|1007|bb|000|084|60|75|7d|3f|e2|67.80|218|2301|12|3084|2470|218|237|744.3593|
  Kotva 0006 prijala signál z tagu 1007. Správa má sekvenčné číslo 084 a patrí do skupiny označenej sekvenčným číslom 000. Správa o prijatí signálu vyslaného tagom bola zaznamenaná v čase definovanom hodnotou čítača 60757d3fe2. Sila prijatého signálu je určená hodnotami 67.80, 218, 2301, 12, 3084, 2470, 218, 237, 744.35937;
- odpoveď na synchronizáciu
  |0006|A|0001|aa|034|136|61|15|93|d9|b5|67.80|184|1888|12|1967|1693|269|240|744.2812|
  Kotva 0006 prijala žiadosť o synchronizáciu od kotvy (master) 0001. Správa má sekvenčné číslo 136 a patrí do skupiny označenej sekvenčným číslom 034. Prijatie žiadosti o synchronizáciu bolo prijaté v čase definovanom hodnotou čítača 611593d9b5. Sila prijatého signálu je určená hodnotami 67.80, 184, 1888, 12, 1967, 1693, 269, 240, 744.28125.

Každá kotva má vlastný čítač, ktorý určuje jej čas. Čas na kotve začne plynúť hneď po jej zapojení do elektrickej siete. Keďže nie je možné zaručiť, že všetky kotvy budú napojené v jeden okamih, treba kotvy vzájomne zosynchronizovať. Synchronizácia prebieha každých n sekúnd. V systéme SEWIO sú to dve sekundy. Keby bol interval dlhší, napr. 10 sekúnd, tak by mohlo dochádzať k dlhým výpadkom lokalizácie, ak by nastala kolízia správ z tagov (tagy nie sú synchronizované) a synchronizačných správ. Naopak pri nízkej perióde synchronizácie (napr. 500 *ms*) by sa zbytočne zarušovalo pásmo na vysielanie.

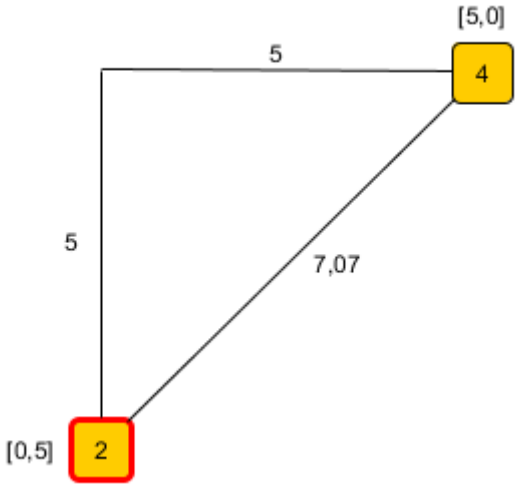
Synchronizácia časov na kotvách sa vzťahuje k času mastera (môže ich byť v systéme viac). Jedná sa o relatívnu synchronizáciu. Master pošle synchronizačnú správu na všetky kotvy. Tým iniciuje synchronizáciu. Na obrázku 3.1 je táto správa znázornená červenými šípkami. Je to správa, v ktorej žiada o aktuálny čas na jednotlivých kotvách. Všetky kotvy, vrátane mastera, pošlú na server čas, v ktorom správu prijali. Takto server získa rovnaký čas na kotvách.

Na serveri sa získaný časový údaj musí ešte upraviť o čas, ktorý bol potrebný na prenos signálu z mastera na jednotlivé kotvy. Čas letu signálu sa spočíta na základe známeho vzorca pre rýchlosť:

$$v = \frac{s}{t} \quad \rightarrow \quad t = \frac{s}{v}$$

[2] Hodnota konštanty je 63 897 600 000

Predpokladáme, že signál letí priamo od mastera k danej kotve. Polohy všetkých kotiev sú známe a teda sa dá jednoducho zistiť aj vzájomná vzdialenosť medzi kotvami. Túto vzdialenosť podelíme rýchlosťou, ktorou sa šíri signál vo vzduchu, t. j. približne rýchlosť svetla ($3 \times 10^8\ m \cdot s^{-1}$) a tým sa získa čas letu signálu medzi dvoma kotvami. Teraz má server k dispozícii referenčný čas pre každú kotvu.



Obr. 3.3: Výpočet vzdialenosti medzi dvoma kotvami (vzdialenosť je udávaná v metroch)

Napríklad na obr. 3.3 sú dve kotvy – označené číslicami 2 a 4. Na základe ich polohy (súradníc) vieme ich horizontálnu a vertikálnu vzdialenosť. Vznikne nám pomyselný pravý trojuholník a pomocou Pytagorovej vety vieme spočítať vzdialenosť (čiže preponu). Na obr. 3.3 sú vzdialenosti v metroch. Čas letu signálu z kotvy 2 na kotvu 4 sa teda spočíta:

$$s = \sqrt{(x_2 - x_4)^2 + (y_2 - y_4)^2} = \sqrt{(0-5)^2 + (5-0)^2} = \sqrt{50} = 7{,}07\ m$$

$$t = \frac{s}{c} = \frac{7{,}07\ m}{3 \times 10^8\ m.s^{-1}} = 2{,}356 \times 10^{-8}\ s$$

Problémom je tiež, že procesor na každej kotve nemá identickú taktovaciu frekvenciu. Dokonca frekvencia v čase kolíše. Je to spôsobné výrobou (v procesoroch sú použité lacné oscilátory). A teda nie je možne mať dva úplne rovnaké procesory. To má za následok, že dve sekundy na jednej kotve nie je rovnako dlhá doba na kotve inej. Teda na inej kotve s inou, aj keď len nepatrne odlišnou, taktovacou frekvenciou procesora môže ísť čas rýchlejšie. A preto treba zosynchronizovať aj samotné hodnoty času poslané kotvami.

Vypočíta sa koeficient, ktorý určí, v akom pomere sú časové intervaly medzi masterom (t_{kotva}) a ostatnými kotvami (t_{master}). Časový interval sa získa ako rozdiel časov po sebe nasledujúcich synchronizačných správ. Koeficient je vlastne pomer dĺžky časového intervalu medzi mastrom a kotvou:

$$koeficient = \frac{t_{kotva}}{t_{master}}$$

Tento koeficient určuje koľkokrát rýchlejšie, respektíve pomalšie, beží čas na danej kotve oproti masteru. Keďže synchronizácia prebieha každých *n* sekúnd, tak aj koeficient sa počíta každých *n* sekúnd. Rovnakým spôsobom je možné vypočítať aj koeficient pre mastera, ktorého hodnota bude

vždy rovná jednej, pretože sa jedná o podiel rovnakých hodnôt. Pre príklad z obr. 3.3 sa koeficienty spočítajú (k_2 je koeficient pre kotvu ② a k_4 je koeficient pre kotvu ④):

$$k_2 = \frac{t_{K2}^1 - t_{K2}^0}{t_{K2}^1 - t_{K2}^0}$$

$$k_4 = \frac{t_{K4}^1 - t_{K4}^0}{t_{K2}^1 - t_{K2}^0}$$

kde pre k_4 (resp. k_2) to je podiel rozdielov časových okamihov prijatia aktuálnej synchronizačnej správy (označeným horným indexom 1) a predchádzajúcej synchronizačnej správy (označeným horným indexom 2).

3.2 Výpočet polohy tagu

V predchádzajúcej podkapitole som popísal synchronizáciu času na kotvách. Táto synchronizácia predchádza situáciám popísaným v tejto kapitole.

Výpočet polohy tagu je implementovaný pomocou metódy TDOA. Z možností popísaných v kapitole 2.1 si firma SEWIO vybrala metódu TDOA, pretože pri tejto metóde nemusí prebiehať komunikácia medzi tagom a kotvami. Tag iba v pravidelných intervaloch vysiela signál. Vďaka tejto minimálnej aktivite tagu sa šetrí batéria a tým pádom sa zvyšuje aj jej výdrž.

Poloha tagu sa počíta v dvojrozmernom priestore. Tag vyšle signál na všetky kotvy vzduchom (modré šípky na obr. 3.1). Kotvy správu prijmú v nejakom čase. Daný čas následne kotvy pošlú do servera, kde sa spočíta poloha tagu.

Server vypočíta rozdiely časov príchodu signálu z tagu medzi rôznymi dvojicami kotiev. Tým získa časový rozdiel príchodu správy medzi danou dvojicou. Na základe tohto rozdielu spočíta hyperbolu, na ktorej sa tag môže nachádzať.

Ako už bolo spomínané hyperbola je kužeľosečka, pre ktorú platí, že absolútna hodnota rozdielu vzdialeností každého jej bodu od dvoch pevne daných bodov je vždy rovnaká. Na obr. 2.2 je znázornená hyperbola s ohniskami F_1 a F_2. V systéme SEWIO sú tieto ohniská vlastne kotvy. Vďaka časovému rozdielu príchodu signálu z tagu na jednotlivé kotvy, server spočíta výslednú hyperbolu.

Pre lepšiu predstavu uvažujme nad systémom s troma kotvami a jedným tagom:

- Tag T so súradnicami x, y;
- Kotvy;
  - K1 so súradnicami X_1, Y_1, táto kotva je master;
  - K2 so súradnicami X_2, Y_2;
  - K3 so súradnicami X_2, Y_3.

Prvým krokom je získanie časového okamihu príchodu signálu vyslaného tagom na jednotlivých kotvách:

$$t_{K1} = k_1 \cdot T_{K1} - k_1 \cdot t_1$$

$$t_{K2} = k_2 \cdot T_{K2} - (k_2 \cdot t_2 - TOF_{K2})$$

$$t_{K3} = k_3 \cdot T_{K3} - (k_3 \cdot t_3 - TOF_{K3})$$

kde t_{Kx} je normalizovaný časový okamih príchodu signálu, k_x je koeficient, T_{Kx} čas prijatia signálu z tagu, t_x je čas poslania synchronizačnej správy a TOF_{Kx} je čas letu signálu z danej kotvy Kx na mastera. V tomto prípade $x \in \{1,2,3\}$. Ďalej sa spočítajú časové rozdiely medzi kotvami:

$$t_{12} = t_{K1} - t_{K2}$$

$$t_{13} = t_{K1} - t_{K3}$$

$$t_{23} = t_{K2} - t_{K3}$$

Vzdialenosť medzi tagom a kotvou sa spočíta na základe vzorca pre Euklidovu vzdialenosť:

$$d_i = \sqrt{(X_i - x)^2 + (Y_i - y)^2}$$

kde index i označuje príslušnú kotvu. Z týchto informácií sa zostaví sústava rovníc, z ktorých sa získajú súradnice tagu.

$$d_{ij} = c \cdot t_{ij} = d_i - d_j = \sqrt{(X_i - x)^2 + (Y_i - y)^2} - \sqrt{\left(X_j - x\right)^2 + \left(Y_j - y\right)^2}$$

kde i, jsú indexy označujúce jednotlivé kotvy a c je rýchlosť signálu. Keďže d_{ij} nie je priamo zmeraná vzdialenosť, ale len odhadnutá na základe rýchlosti signálu a času letu signálu, používa sa pre d_{ij} pojem pseudovzdialenosť. Takže rovnice pre kotvy K1, K2, K3 budú vyzerať:

$$d_{12} = c \cdot t_{12} = d_1 - d_2 = \sqrt{(X_1 - x)^2 + (Y_1 - y)^2} - \sqrt{(X_2 - x)^2 + (Y_2 - y)^2}$$

$$d_{13} = c \cdot t_{13} = d_1 - d_3 = \sqrt{(X_1 - x)^2 + (Y_1 - y)^2} - \sqrt{(X_3 - x)^2 + (Y_3 - y)^2}$$

$$d_{23} = c \cdot t_{23} = d_2 - d_3 = \sqrt{(X_2 - x)^2 + (Y_2 - y)^2} - \sqrt{(X_3 - x)^2 + (Y_3 - y)^2}$$

Teraz máme rovnicu troch sústav o dvoch neznámych x a y. Riešením tejto sústavy sú súradnice tagu.

4 Kalmanov filter

Kalmanov filter je algoritmus, ktorý používa sadu nameraných hodnôt v čase a statický šum pre odhad nasledujúcej neznámej hodnoty. Je pomenovaný podľa matematika a elektrotechnického inžiniera Rudolfa Emil Kálmána, ktorý vyvinul tento filter (algoritmus).

Kalmanov filter má široké využitie v rôznych technologický aplikáciách. Najrozšírenejšie použitie Kalmanovho filtra je v navigácií vozidiel, prípadne v letectve a vo vesmíre. Kalmamov filter používa na „vyhladenie“ trasy aj GPS. Vzhľadom k rôznym faktorom, ako sú napríklad interferencie GPS signálu, by trasa vozidla v čase oscilovala v okolí jeho reálnej polohy. Kalmanov filter tieto odchylky (šum) odstraňuje. Samozrejme GPS navigácie nepoužívajú na korekciu trasy len Kalmanov filter, ale tento filter je veľmi dôležitou súčasťou týchto optimalizácií. Ďalšie využitie Kalmanovho filtra je v oblasti spracovania signálov alebo v ekonometrii [8].

Dá sa povedať, že úlohou Kalmanovho filtra je odhadnúť hodnotu v nasledujúcom kroku (meraní).

Keďže Kalmanov filter nepoužíva priamo minulú hodnotu odhadu, ale je tam jeden medzikrok, výpočet predpovede, tak pri niektorých príležitostiach sa používajú pojmy

- *a priori* odhad – pre predpovedanú hodnotu;
- *a posteri* odhad – pre odhadovanú hodnotu.

Nasledujúce informácie kapitoly 4 som čerpal zo zdroja [9].

4.1 Algoritmus Kalmanovho filtra

Aj keď v názve sa nachádza slovo „filter“, lepšie je považovať Kalmanov filter za algoritmus. Je to rekurzívny algoritmus, vykonávajúci sekvenčne štyri kroky. Ako vstup prijíma jednu hodnotu z_k (nameraná hodnota) a vracia jeden výstup $\hat{x}_k$ (odhadnutá hodnota). Na obr. 4.1 je daný algoritmus zobrazený v sivom obdĺžniku ohraničenom prerušovanou čiarou.

V popise algoritmu sa nachádza akcent „-“ pri niektorých premenných ako horný index. Toto označenie je veľmi dôležité. Označuje predpovedanú hodnotu. Jednotlivé kroky z obr. 4.1 sa dajú jednoducho popísať nasledovne:

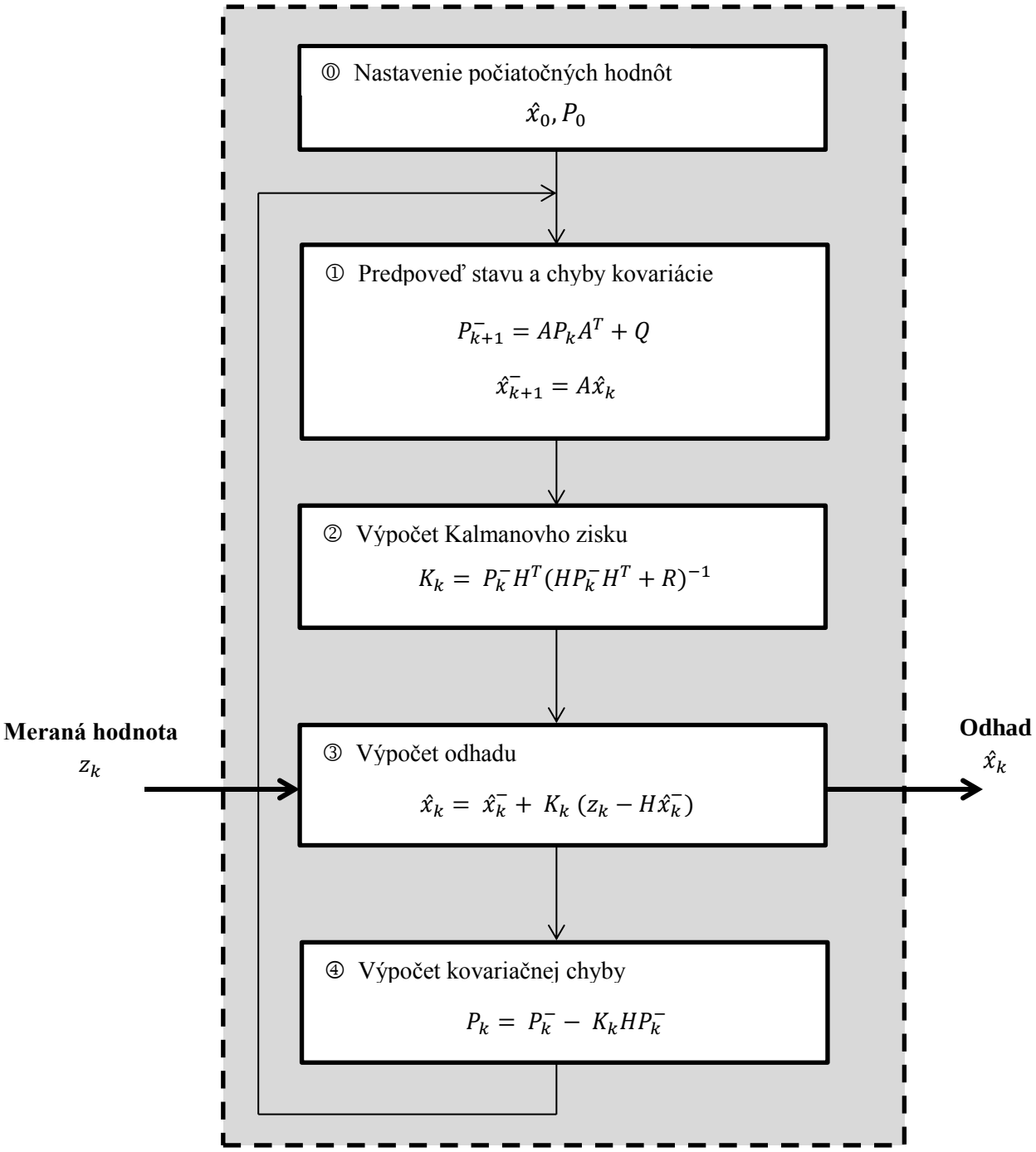
⓪ Nastavenie počiatočných hodnôt;

① Predpoveď. Výpočet predpovede ($\hat{x}_k^-$) a predpovede kovariačnej chyby (P_k^-) pre nasledujúcu časovú hodnotu. Premenné $\hat{x}_k^-$ a P_k^- obsahujú spomínaný horný index;

② Výpočet Kalmanovho zisku K_k (*angl. Kalman gain*). Premenná P_k^- je spočítaná v predchádzajúcom kroku a premenné *H* a *R* sú nastavené mimo algoritmu;

③ Odhad nasledovnej hodnoty je spočítaný zo vstupu;

④ Výpočet kovariačnej chyby. Kovariačná chyba indikuje, ako presný je daný odhad.



Obr. 4.1: Algoritmus Kalmanovho filtra [9]

V tabuľke 4.1 sú zobrazené všetky prvky vystupujúce v algoritme Kalmanovho filtra. Premenné označené ako „Systémový model“ sú nastavené pred výpočtom algoritmu. Inými slovami, tieto premenné majú statickú (stále rovnakú) hodnotu a teda nie sú počítané v Kalmanovom filtri. Musia byť nastavené užívateľom vzhľadom na charakteristiku systému a účelu Kalmanovho filtra. Tieto premenné určujú výkonnosť (presnosť) Kalmanovho filtra. Čím lepšie popisujú systém, ktorý modelujú, tým lepšie funguje Kalmanov filter. Vzťah medzi Kalmanovým filtrom a systémovým modulom bude popísaný v kapitole 4.1.3. Kalmanov filter pracuje s vektormi a maticami. Vektormi sú premenné:

- z_k – nameraná hodnota;
- $\hat{x}_k$ – odhadnutá hodnota;
- $\hat{x}_k^-$ – predpovedaná hodnota.

Maticami sú premenné:

- A – popisuje daný systém;
- H – vzťah medzi meraním a stavovou premennou;
- Q – šum počas prenosu;
- R – šum počas merania;
- K_k – Kalmanov zisk;
- P_k – odhad kovariačnej chyby;
- P_k^- – predpoveď kovariačnej chyby.

Vstup algoritmu	z_k
Výstup algoritmu	$\hat{x}_k$
Systémový model	A, H, Q, R
Pre vnútorný výpočet	$\hat{x}_k^-, P_k^-, P_k, K_k$

Tab. 4.1: Popis premenných Kalmanovho filtra

Na obrázku 4.1 je Kalmanov filter rozdelený do štyroch krokov. Kalmanov filter je taktiež možné rozdeliť na dve časti vzhľadom na význam daných krokov. Sú to proces predikcie a proces estimácie.

- **Proces predikcie (*angl. Prediction process*)**
  Do tohto procesu spadá krok ① z obr. 4.1. Odhad a kovariačná chyba z predchádzajúceho časového okamihu ($\hat{x}_{k-1}$, P_{k-1}) sú vstupom predpovede týchto dvoch hodnôt v aktuálnom čase ($\hat{x}_k^-$, P_k^-) a vrátené ako výsledok. Tieto hodnoty sú použité v estimačnom procese.
- **Proces estimácie (*angl. Estimation process*)**
  Tento proces zahŕňa kroky ②, ③, ④. Výsledkom estimačného procesu sú odhad ($\hat{x}_k$) a kovariačná chyba (P_k). V tomto procese je tiež zahrnutá meraná hodnota ako vstup a pridaná k predpovedaným hodnotám z predchádzajúceho procesu.
  Cieľom je teda vypočítať odhad, ktorý je vlastne výsledkom Kalmanovho filtra.
  V ďalších kapitolách podrobnejšie popíšem procesy predikcie a estimácie a systémový model.

4.1.1 Proces predikcie

Tento proces spadá pod prvý krok z obr. 4.1. Je jednoduchší ako estimačný proces, keďže pozostáva len z jedného kroku. Predpovedá, akú hodnotu bude mať odhad v nasledujúcom časovom okamih z aktuálneho časového okamihu. Rovnice z prvého kroku Kalmanovho filtra sú:

$$\hat{x}_{k+1}^{-} = A\hat{x}_k \tag{4.1}$$

$$P_{k+1}^{-} = AP_kA^T + Q \tag{4.2}$$

Hodnoty $\hat{x}_k$ a P_k sú spočítané v krokoch ③, resp. ④. Premenné A a Q sú dopredu definované modelom systému. Môžeme si všimnúť, že v tomto procese sú použité len premenné A a Q zo systémového modelu. V estimačnom procese zase naopak sú použité len premenné H a R zo systémového modelu. Dolný index „*k+1*“ indukuje hodnotu z nasledujúceho časové ho okamihu t_{k+1}. Pre lepšiu orientáciu je pridaná nasledovná tabuľka.

$\hat{x}_k$	Odhad stavovej premennej
$\hat{x}_k^-$	Predpoveď stavovej premennej
P_k	Odhad kovariačnej chyby
P_k^-	Predpoveď kovariačnej chyby

Tab. 4.2: Rozdiely v zápise odhadovaných a predpovedaných hodnôt

Rozdiel medzi predpoveďou a odhadom

Znovu sa pozrime na rovnicu výpočtu odhadu:

$$\hat{x}_k = \hat{x}_k^- + K_k\,(z_k - H\hat{x}_k^-) \tag{4.3}$$

V tejto rovnici 4.3 je potrebné si uvedomiť, že $\hat{x}_k^-$, čize predpovedaná hodnota, sa spočíta na základe predchádzajúceho odhadu (rovnica 4.1). Po dosadení za premennú $\hat{x}_k^-$ je jednoznačne vidieť, že predpovedaná a odhadovaná hodnota nie sú to isté:

$$\hat{x}_k = A\hat{x}_{k-1} + K_k(z_k - HA\hat{x}_{k-1})$$

4.1.2 Proces estimácie (odhadu)

Prvou časťou procesu estimácie je výpočet Kalmanovho zisku (krok ②). Kalmanov zisk sa používa na váhovanie výpočtu odhadu. Výpočet Kalmanovho zisku:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

V nasledujúcom kroku je spočítaný odhad ($\hat{x}_k$), ktorý je výstupom algoritmu. Odhadnutá hodnota je získaná súčtom predpovedanej hodnoty z kroku ① a chybou predpovedanej hodnoty váhovanou Kalmanovým ziskom. Chyba je získaná ako rozdiel nameranej hodnoty a predpovedanej hodnoty.

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

Z estimačného procesu zostáva už len posledný krok, štvrtý, ktorý počíta kovariačnú chybu. Štvrtý krok sa spočíta nasledovne:

$$P_k = P_k^- - K_kHP_k^-$$

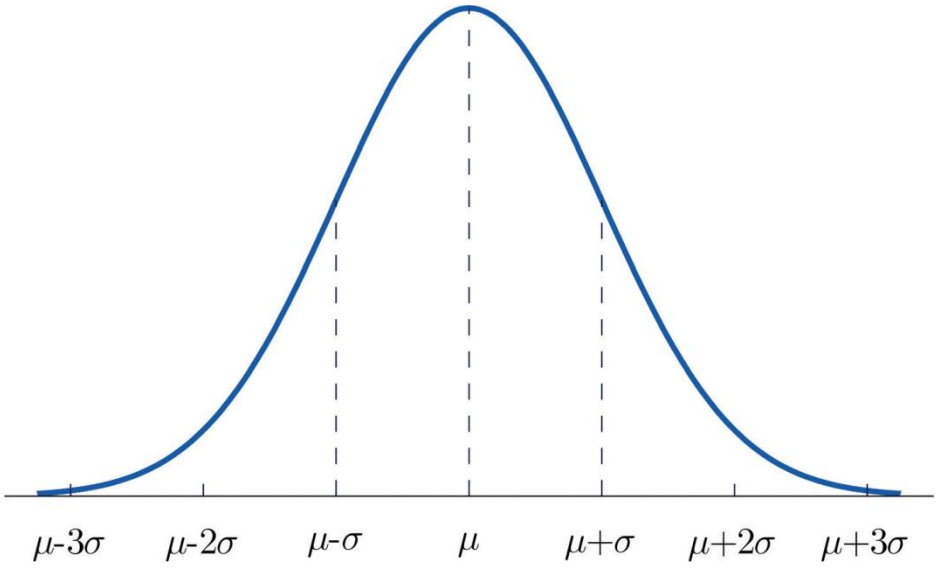
Kovariačná chyba je rozdiel medzi odhadom spočítaným Kalmanovým filtrom a skutočnou, ale neznámou, hodnotou. Neznámou pretože Kalmanov filter odhaduje nasledujúcu hodnotu. Preto skutočná hodnota, ku ktorej sa odhad vzťahuje, príde na vstup Kalmanovho filtra až v ďalšom časovom okamihu. Inými slovami sa dá povedať, že kovariačná chyba je stupňom presnosti daného odhadu.

Pre vzťah medzi kovariačnou chybou a odhadom platí priama úmernosť. Ak je kovariačná chyba veľká, chyba odhadu je taktiež veľká. A naopak, ak je kovariačná chyba malá, tak je mála aj chyba odhadu. Vzťah medzi reálnou vzorkou (x_k), jej odhadom ($\hat{x}_k$) a kovariačnou chybou sa dá definovať pomocou normálneho rozloženia (*angl. Normal distribution*):

$$x_k \sim N(\hat{x}_k, P_k)$$

Z tejto rovnice vyplýva, že odhad Kalmanovho filtra a jeho kovariačná chyba sú hodnoty vyjadrené normálnym rozložením premennej x_k. Grafom normálneho rozloženia je Gaussova krivka. Jej šírku určuje P_k (v tomto prípade kovariačná chyba). A tu je vysvetlená priama úmernosť medzi odhadom a kovariačnou chybou spomenutou vyššie.

Keď je šírka krivky normálneho rozloženia malá, x_k môže nadobúdať hodnoty blízke strednej hodnote. V našom ponímaní je stredná hodnota odhad Kalmanovho filtra. Na druhej strane, ak je šírka krivky veľká, rozsah hodnôt, ktoré môže x_k nadobúdať, je väčší. A to robí väčšiu aj chybu odhadu. Na obr. 4.2 je zobrazený graf normálneho rozloženia.



Obr. 4.2: Graf normálneho rozloženia[18]

4.1.3 Systémový model

Pod týmto pojmom je myslené „matematické vyjadrenie daného problému“. Ak je takýto model známy, nie je problém k nemu implementovať Kalmanov filter. Avšak modelovanie systému matematicky je dosť ťažká úloha.

Kalmanov filter pracuje s lineárnym stavovým modelom popísaným nasledovnými rovnicami:

$$x_{k+1} = Ax_k + w_k \tag{4.4}$$

$$z_k = Hx_k + v_k \tag{4.5}$$

V daných rovniciach je dôležité, že do modelu je zanesený aj šum. Význam jednotlivých premenných je popísaný v nasledujúcej tabuľke.

x_k	stavová premenná	stĺpcový vektor	n × 1
z_k	meranie	stĺpcový vektor	m × 1
A	prechodová stavová matica	matica	n × n
H	vzťah stavu a merania	matica	m × n
w_k	prechodový stavový šum	stĺpcový vektor	n × 1
v_k	šum v meraní	stĺpcový vektor	m × 1

Tab. 4.3: Význam premenných systémového modelu

Stavová premenná je fyzikálna veličina, napr. pozícia, rýchlosť, hmotnosť, teplota, a i. Dôležitú úlohu v Kalmanovom filtri zohráva šum. Všetok šum je považovaný za biely šum (rovnomerne pokrýva všetky frekvencie). Prechodový stavový šum (*angl. state transition noise*) w_k je zavedený do systému a vplýva na stavovú premennú a šum v meraní v_k je šum ktorý vznikol na senzore.

Matice *A* a *H* majú všetky prvky konštantné. Matica *A* popisuje zmeny systému v čase. Matica *H* popisuje vzťah medzi meraním a stavovou premennou. Definuje ako je namapovaná stavová premenná na meranie.

Teraz sa pozrime na časti Kalmanovho filtra, ktoré sú relevantné k systémovému modelu. Sú to dve časti. Prvou je výpočet predpovede z odhadu. A je odvodená od rovnice 4.4 systémového modelu.

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

Druhá časť výpočtu odhadu z tretieho kroku Kalmanovho filtra, ktorá vychádza z rovnice 4.5, je označená v nasledovnom vzťahu hrubým písmom.

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

Porovnaním týchto dvoch rovníc a rovníc systémového modelu 4.4 a 4.5 je zrejmé, že jediný rozdiel je neprítomnosti šumu. Matica *H* je použitá aj na inom mieste výpočtu, ale tam neplní takú dôležitú úlohu vzhľadom na model systému ako v zobrazených rovniciach.

Kovariačný šum

Vo všeobecnosti šum je hodnota, ktorá sa nedá predpovedať, môžeme ju len štatisticky určiť. V Kalmanovom filtri máme vždy šum s normálnym rozložením hodnôt, t. j. stredná hodnota je vždy nulová. Šum v Kalmanovom filtri je vyjadrený nasledovnými kovariačnými maticami. Obe sú to diagonálne matice (majú nulové hodnoty mimo hlavnú diagonálu).

Q	kovariačná matica pre $\boldsymbol{w_k}$	n × n
R	kovariačná matice pre $\boldsymbol{v_k}$	m × m

Tab. 4.4: Kovariačné matice pre šum

Kovariačná matica je definovaná ako matica obsahujúca zmeny (variácie) premennej. Napríklad máme *n* šumov w_1, w_2, ..., w_n a zmena každej premennej je definovaná ako $\sigma_1^2,\ \sigma_2^2, \sigma_n^2$, tak kovariačná matica Q môže byť zapísaná takto:

$$Q = \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_n^2 \end{bmatrix}$$

Kovariačná matica *R* sa definuje ekvivalentne. Existuje *m* šumov v_1, v_2, ..., v_m a zmena každej premennej je definovaná ako $\vartheta_1^2,\ \vartheta_2^2, \vartheta_m^2$, tak kovariačná matice *R* môže byť zapísaná takto:

$$R = \begin{bmatrix} \vartheta_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \vartheta_m^2 \end{bmatrix}$$

Je nevyhnutné, aby boli matice Q a R definované čo najpresnejšie. Zároveň analytické určenie matíc nie je možné kvôli rôznym zdrojom šumu. Preto je treba určiť matice empiricky, t. j. sériou pokusov a omylov.

Nie je ale úplne vhodné začať skúšať bez prípravy. Keď sa pozrieme, kde sú matice Q a R umiestnené v Kalmanovom filtri, môže vypozorovať určité vlastnosti, ktoré nám môžu pomôcť s určením hodnôt v matici.

Začneme s maticou *R* a nasledujúcou rovnicou 4.6 z druhého kroku Kalmanovho filtra.

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \tag{4.6}$$

Predpokladajme, že všetky hodnoty v rovnici 4.6 sú skalárne. Potom môžeme túto rovnicu vyjadriť nasledovne (maticový zápis je ponechaný kvôli lepšej zrozumiteľnosti pri porovnaní s predchádzajúcou rovnicou 4.6).

$$K_k = \frac{P_k^- H^T}{H P_k^- H^T + R} \tag{4.7}$$

Keďže matica *R* sa nachádza v menovateli, tak zvyšovaním hodnoty *R* (zvyšovaním je myslené zvyšovanie hodnôt na diagonále matice *R*) prichádza k znižovaniu hodnoty Kalmanovho zisku. Aby sme zistili, čo znamená znižovanie Kalmanovho zisku, pozrime sa znova na túto rovnicu pre výpočet odhadu:

$$\hat{x}_k = (1 - K_k)\hat{x}_k^- + K_k z_k \tag{4.8}$$

Keď klesá Kalmanov zisk odchýlka odhadu klesá, pretože odhad je menej ovplyvňovaný externým meraním. Preto ak chceme získať odhad s menšou odchýlkou, treba zvyšovať *R*.

Podobne sa zameriame aj na maticu *Q*. Tá sa používa v predpovedi kovariačnej chyby (rovnica 4.2).

$$P_{k+1}^- = A P_k A^T + Q$$

Tu je situácia o trochu jednoduchšia. Ak rastie predpoveď kovariačnej chyby, tak rastú aj hodnoty na diagonále matice Q. Znovu si vezmime rovnicu 4.7 a považujme všetky jej premenné za sklárne. P_k^- sa nachádza v čitateli aj v menovateli, ale čitateľ rastie o niečo rýchlejšie vďaka pričítaniu

hodnoty R v menovateli. Takže ak rastie hodnota Q (momentálne pre zjednodušenie ju považujem za skalárnu), rastie aj Kalmanov zisk. Keď rastie Kalmanov zisk, rastie aj prírastok z meraniu (rovnica 4.8).

Teda, ak chceme mať menšiu odchýlku pri odhade, treba znižovať hodnoty na diagonále matice Q. Je to presný opak, ako s upravovať hodnoty na diagonále matice *R*.

4.2 Implementácia Kalmanovho filtra

Zrejme najznámejším využitím Kalmanovho filtra je pre odhad pozície a rýchlosti objektu. Podobný princíp, aj keď s viacerými korektúrami sa používa aj v GPS navigáciách (v navigáciách sú mapy a teda sa dá predpokladať, že auto pôjde po ceste, resp. jeho trajektória nebude viesť cez domy a i.).

4.2.1 Teoretický príklad odhadu pozície a rýchlosti Kalmanovým filtrom

Prvým krokom je určenie systémového modelu. Zaujímame sa o rýchlosť a pozíciu, takže stavovou premennou x bude vektor obsahujúci pozíciu (*angl. position*) a rýchlosť (*angl. velocity*):

$$x = \begin{Bmatrix} position \\ velocity \end{Bmatrix}$$

Model, ktorý popisuje danú situáciu je definovaný:

$$x_{k+1} = Ax_k + w_k$$

$$z_k = Hx_k + v_k$$

Význam premenných je popísaný v kapitole 4.1.3. Premenné A a H sú matice definované nasledovne:

$$A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Kde premenná Δt časový interval medzi meraniami pozície. Aby bolo vidieť, že daný systém je správny, vyjadrím premenné popisujúce stavovú premennú.

$$x_{k+1} = Ax_k + w_k$$

$$\begin{bmatrix} position \\ velocity \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} position \\ velocity \end{bmatrix}_k + \begin{bmatrix} 0 \\ w_k \end{bmatrix}$$

$$\begin{bmatrix} position \\ velocity \end{bmatrix}_{k+1} = \begin{bmatrix} position + velocity \cdot \Delta t \\ velocity + w \end{bmatrix}_k$$

Najprv sa pozrieme pre výraz pozície:

$$position_{k+1} = position_k + velocity \cdot \Delta t \tag{4.9}$$

Toto je matematické vyjadrenie fyzikálneho princípu:

- nová pozícia je rovná súčtu predchádzajúcej pozície a posunutia.

Druhý výraz popisuje rýchlosť:

$$velocity_{k+1} = velocity_k + w_k \tag{4.10}$$

V rovnici 4.9 nie je zahrnutý systémový šum w_k, pretože sa jedná o rýdzo matematické vyjadrenie a systémový šum nemá priamy vplyv na pozíciu. V rovnici 4.10 pre rýchlosť je systémový šum, pretože má vplyv na rýchlosť. Napr. ak by sa jednalo o rýchlosť vlaku, tak na rýchlosť vplývajú aj faktory ako trenie, porucha riadiacej jednotky a i.

Druhou rovnicou systémového modelu je rovnica merania (rovnica 4.5). Použitím premennej H získame zápis:

$$z_k = Hx_k + v_k$$

$$z_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} position \\ velocity \end{bmatrix}_k + v_k$$

$$z_k = position_k + v_k$$

Tento zápis hovorí, že meraná bola pozícia objektu. Zároveň bol zahrnutý aj šum v_k, teda šum vzniknutý pri meraní. Z tohto zápisu je možné vidieť, že matica H určuje, ktorú stavová premenná bude meraná. Keby bolo treba merať rýchlosť, tak stačí zmeniť maticu H:

$$H = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

Potom by rovnica merania vyzerala takto:

$$z_k = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} position \\ velocity \end{bmatrix}_k + v_k$$

$$z_k = velocity_k + v_k$$

4.2.2 Kalmanov filter pre výpočet pozície v dvojrozmernom priestore

Predchádzajúci príklad síce názorne ukazuje, ako získať pozíciu alebo rýchlosť z daného modelu, ale prakticky nie je príliš využiteľný. Vzhľadom na zisťovanie polohy nejakého objektu je lepšie mať model, kde bude poloha určená (aspoň) dvoma súradnicami. Taký model vyzerá:

$$x = \begin{Bmatrix} position_x \\ velocity_x \\ position_y \\ velocity_y \end{Bmatrix}$$

Pojem $velocity_x$ znamená rýchlosť v smere osi x. Podobne to platí pre $velocity_y$, kde sa jedná o rýchlosť v smere osi y. Matice A a H sú definované:

$$A = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

A takto definovaný model dokáže odhadnúť súradnice objektu v dvojrozmernom priestore.

4.3 Testovanie funkčnosti Kalmanovho filtra pre dvojrozmerný systémový model

Moju implementáciu Kalmanovho filtra pre dvojrozmerné dáta (ďalej označovaný ako 2D KF) som testoval na vygenerovaných súradniciach. Vygenerované súradnice som použil, aby som mal presné hodnoty a tým pádom som mohol zistiť účinnosť implementácie 2D KF. Vygeneroval som štyri rôzne dráhy objektu:

- Obdĺžnik;
- Šesťuholník;
- Kružnicu;
- Sínusoidu.

Štyri rôzne dráhy som vygeneroval hlavne preto, aby som 2D KF nepretrénoval na jednom druhu dát (napr. obdĺžniku). Vzorkovanie súradníc nie je pre jednotlivé objekty rovnaké. Vychádza to z rôznych tvarov a veľkostí daných objektov. Vzorkovanie je nasledovné:

- Obdĺžnik – je tvorený 240 bodmi. Vzdialenosť medzi dvoma po sebe idúcimi bodmi je vždy presne 10 pixelov;
- Šesťuholník – je tvorený 160 bodmi. V horizontálnych líniách je vzdialenosť medzi bodmi 10 pixelov. V šikmých líniách je vzdialenosť medzi bodmi približne 11 pixelov (11,18 pixelu presne);
- Kružnica – je tvorená 72 bodmi. Vzdialenosť medzi bodmi po kružnici je približne 21 pixelov;
- Sínusoida – je tvorená 81 bodmi a vzdialenosti na nej sa líšia v závislosti na úseku sínusoidy. V na ose *x* je vždy posun o 10 pixelov.

Nie sú nevyhnutné presné dĺžkové jednotky, pri testoch som rátal vzdialenosti v pixeloch. Pre lepšiu predstavu by sa dal jeden pixel považovať za 10 cm, a teda vzdialenosť 10 pixelov by predstavovalo vzdialenosť jeden meter medzi bodmi (napr. v obdĺžniku).

K súradniciam jednotlivých objektov som pripočítal určitý šum (náhodne vygenerovaný Gaussov šum). Tento náhodný šum mal strednú hodnotu rovnú nule a hodnota smerodajnej odchýlky bola 15. Hodnota 15 je určená v pixeloch (príp. 1,5 metra). Taká výrazná odchýlka bola zvolená pre jednoduchú vizuálnu kontrolu na obrázkoch. Tým som zabezpečil „reálne hodnoty“, t.j. hodnoty, ktoré by teoreticky mohli zodpovedať reálne nameraným hodnotám s chybami (napr. pri meraní alebo pri prenose signálu). Tieto hodnoty ovplyvnené Gaussovým šumom som následne poslal 2D KF

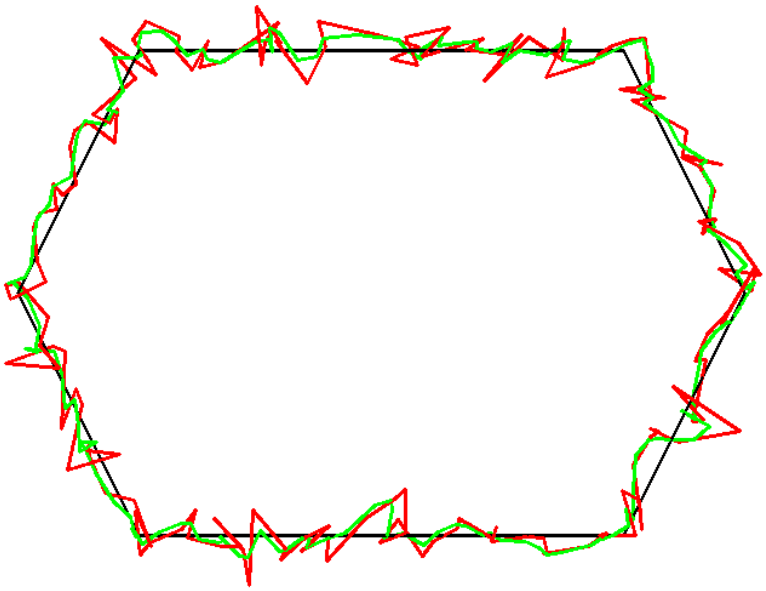
a získal som dráhu, ktorá sa podľa 2D KF mala čo najviac približovať k pôvodnej (vygenerovanej) dráhe.
Výstupom testovania sú:

- Obrázky – zobrazujú vygenerovanú trasu, trasu ovplyvnenú Gaussovým šumom a trasu spočítanú 2D KF;
- Videá – pre zobrazenie výpočtu v reálnom čase (zobrazujú ako sa 2D KF „rozhodoval“ v jednotlivých bodoch)
- Výpočet priemernej chyby vzdialenosti dát ovplyvnených Gaussovým šumom a dát spočítaných 2D KF od vygenerovaných dát. Priemerná chyba vzdialenosti sa spočítala ako súčet vzdialeností bodov ovplyvnených Gaussovým šumom (resp. spočítaných 2D KF) od vygenerovaných bodov v prislúchajúcom časovom kroku.

Hlavným zmyslom testovania 2D KF je hľadanie optimálnej hodnoty matice Q. Ako je písané v kapitole 4.1.3 tieto hodnoty matice sa dajú najlepšie určiť len empiricky. Preto som sa v testovaní sústredil hlavne na konfiguráciu matice Q.

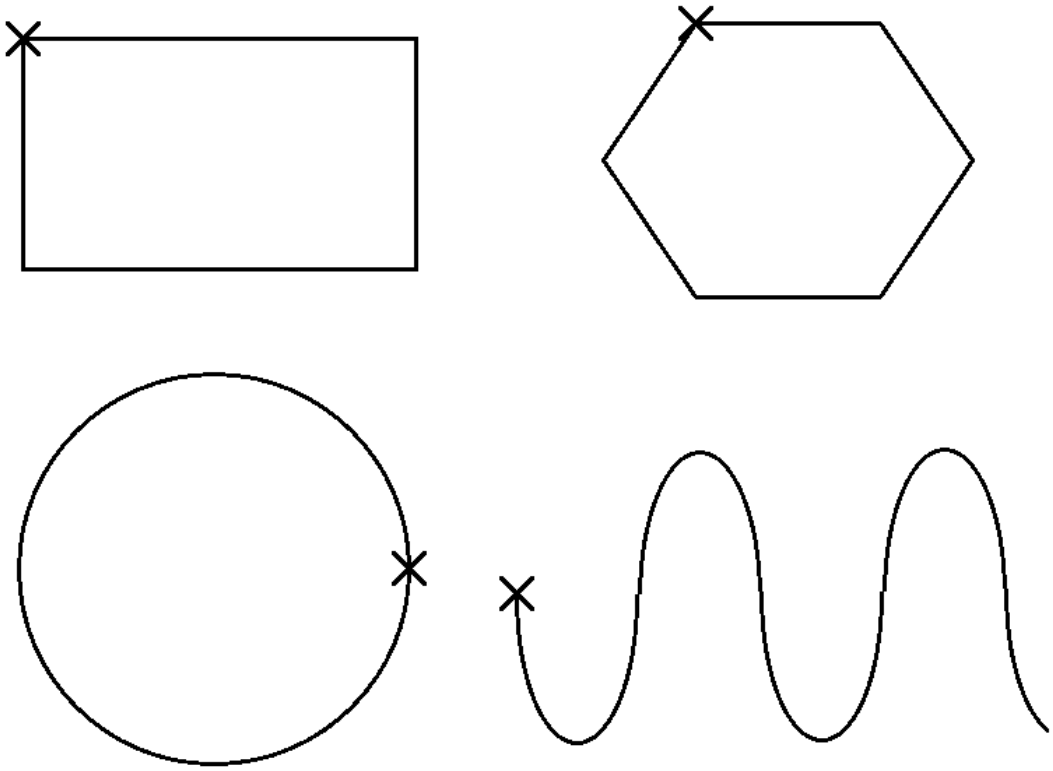
Na obrázkoch a videách (príklad takéhoto obrázku je na obr. 4.3), ktoré zobrazujú výsledky sú jednotlivé trajektórie farebne rozlíšené:

- Čierna farba – pôvodné (vygenerované dáta);
- Červená farba – dáta ovplyvnené Gaussovým šumom;
- Zelená farba – dáta spočítané 2D KF.



Obr. 4.3: Príklad obrázku z troma trajektóriami

Počiatočné súradnice sú pre jednotlivé geometrické útvary zobrazené na obr. 4.4 znakom „×“. Smer danej trajektórie je vždy z smere hodinových ručičiek.



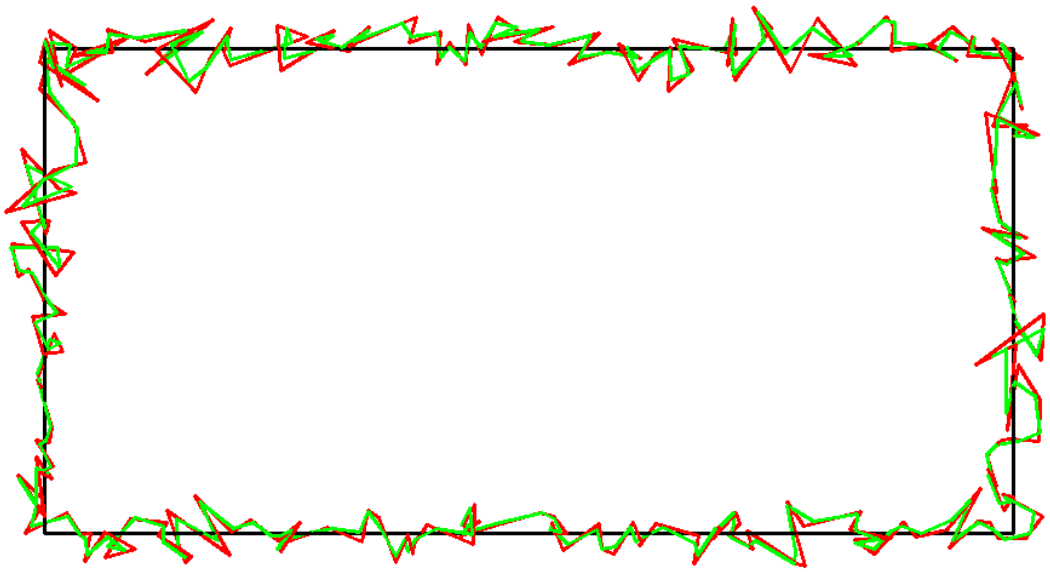
Obr. 4.4 Počiatočné súradnice jednotlivých vygenerovaných útvarov

Konfigurácia matice Q

Zmenou hodnôt na diagonále matice Q (je to diagonálna matica) sa určuje uhladenosť výslednej krivky. Nižšie hodnoty krivku vyhladzujú. Lenže pri nižších hodnotách nie je 2D KF schopný dostatočne rýchlo reflektovať zmenu smeru. Prejaví sa to výrazným vzdialením odhadovanej dráhy od skutočnej dráhy. Takže je potrebné nájsť takú konfiguráciu matice Q, aby bola krivka dostatočne vyhladená a zároveň dokázala reagovať dobre aj na zmenu smeru. Na sérií pokusov teraz ukážem vplyv matice Q na výslednú dráhu spočítanú 2D KF. Začal som s touto maticou:

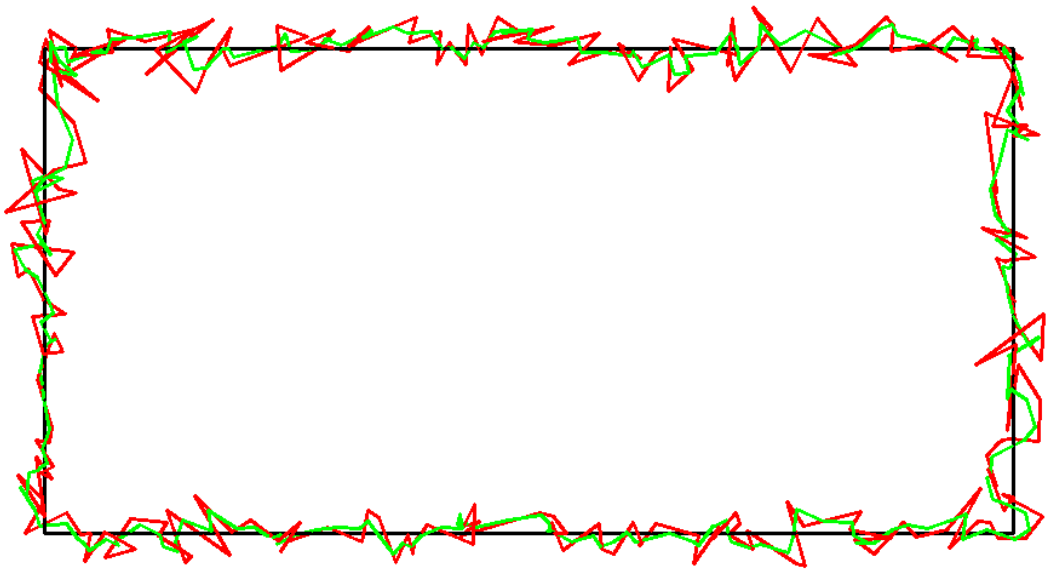
$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$$

Výsledok pri tejto konfigurácií matice je na obr. 4.5.



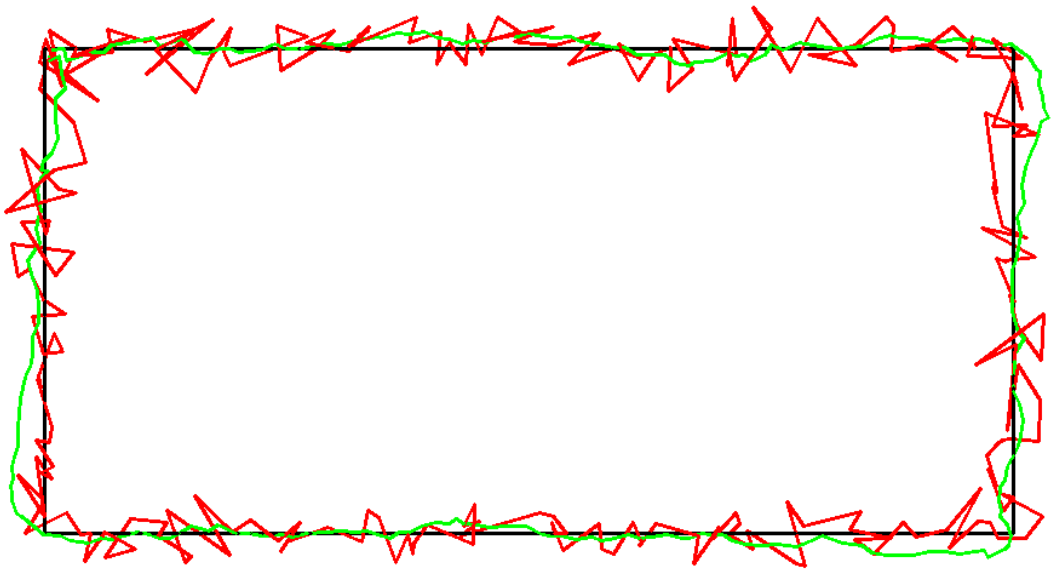
Obr. 4.5: Relatívne vysoké hodnoty (10) na diagonále matice Q

Na obr. 4.5 je vidieť, že 2D KF príliš červenú dráhu nevyhladil. Keďže znižovanie hodnôt na diagonále Q vyhladzuje dráhu 2D KF, tak postupne budem znižovať hodnoty na diagonále. Pre jednoduchosť mám na diagonále vždy tie isté hodnoty. Zo začiatku som zmenšoval hodnoty o dva. Na obr. 4.6 je použitá matica Q s hodnotou jedna na diagonále.



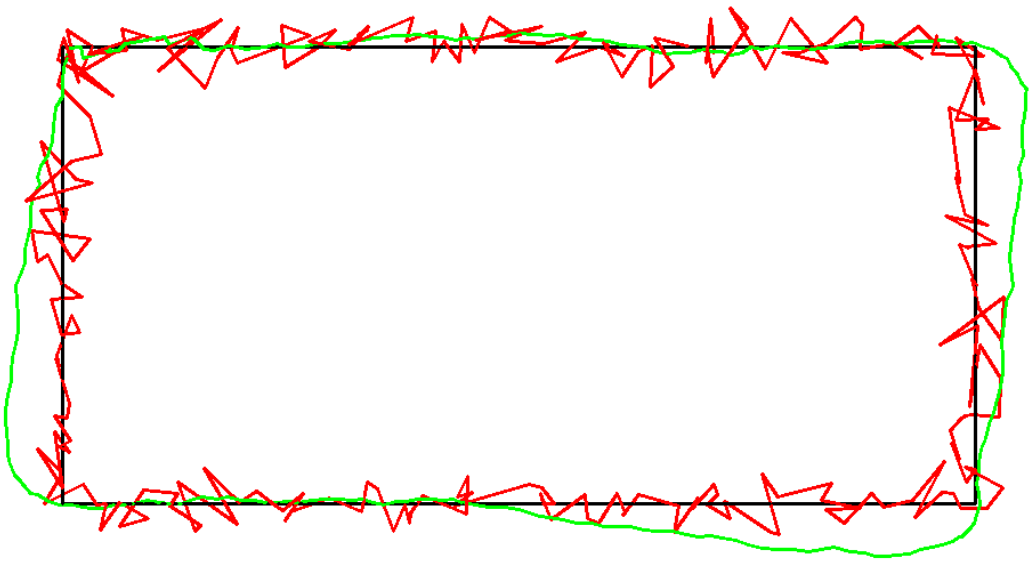
Obr. 4.6: Hodnoty jedna na diagonále Q

Zelená dráha na obr. 4.6 je už lepšia ako na obr. 4.3, ale stále to príliš nepodobá na pôvodný (čierny) obdĺžnik. Preto som ďalej znižoval hodnoty. Na obr. 4.7 sú hodnoty na diagonále Q rovné 0,01.



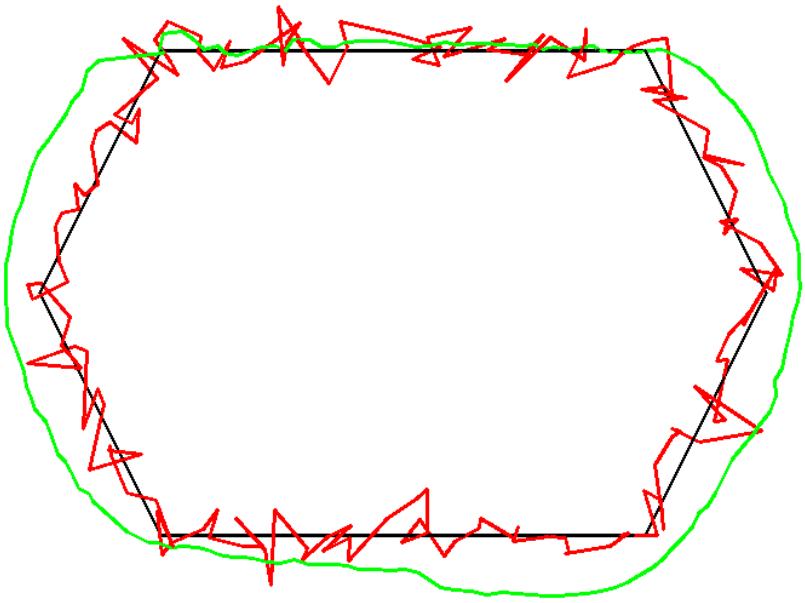
Obr. 4.7: Hodnoty 0,01 na diagonále Q

Pri takto nízkej hodnote je už vyhladenie krivky relatívne dobré. Ukazuje sa však negatívny efekt znižovania hodnôt na diagonále. A tým je slabá reakcia 2D KF na zmenu smeru. Tento jav je možné sledovať vo vrcholoch obdĺžnika (okrem počiatočného vrcholu). Vo vrcholoch obdĺžnika sa mení smer dráhy. 2D KF ale predpokladá, že dráha bude pokračovať, a preto sa zelená trajektória vzdiali od pôvodnej. Potom trvá nejakú dobu (približne 7 bodov na tejto dráhe), kým sa vráti k pôvodnej (čiernej) dráhe. Lepšie je tento jav vidieť, keď sa hodnoty ešte znížia. Na obr. 4.8 sú hodnoty rovné 0,001.

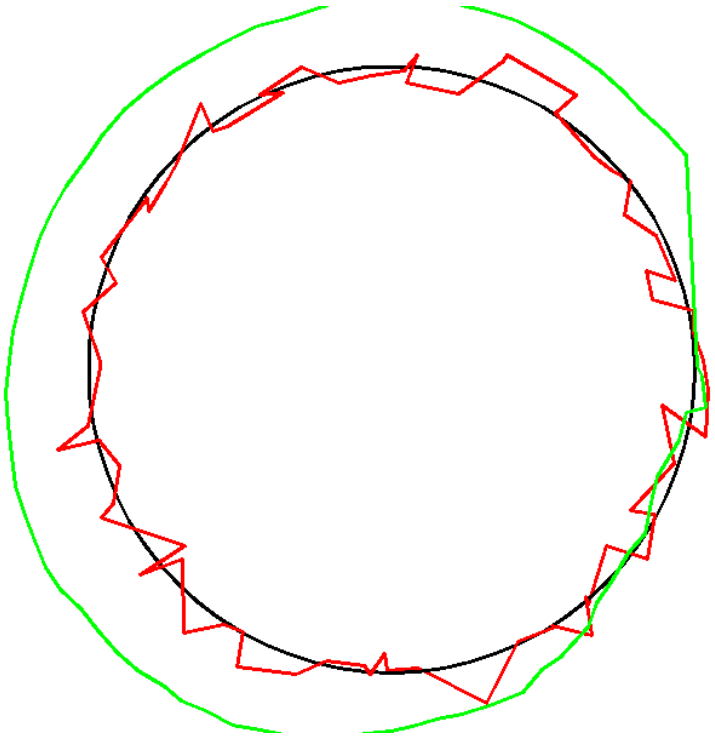


Obr. 4.8: Hodnoty 0,001 na diagonále Q

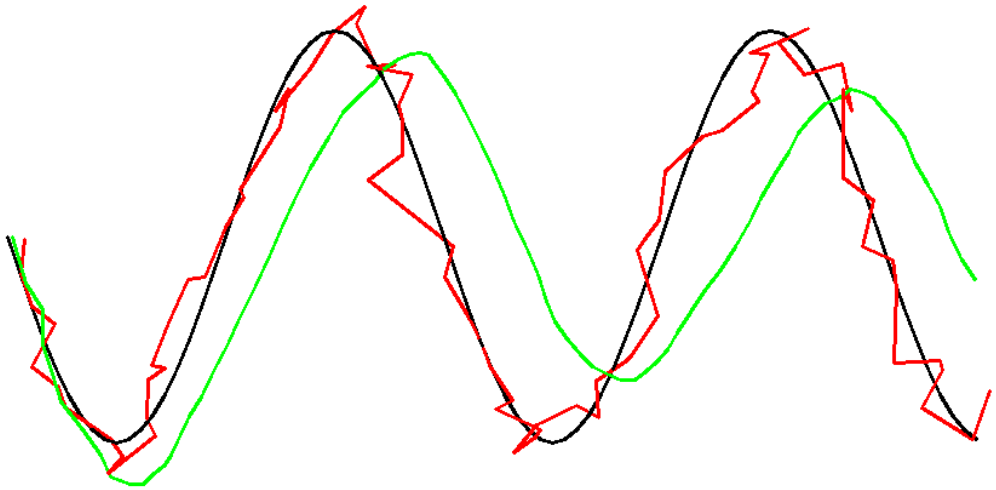
Zdalo by sa, že hodnota 0,01 je v podstate ideálna. Lenže zatiaľ som testoval iba obdĺžnikovú trajektóriu. Preto som sa rozhodol použiť už „nájdenú“ maticu Q a zmenil som tvar trajektórie. Na obr. 4.9, 4.10 a 4.11 sú ostatné tvary trajektórií, pričom matica Q zostala rovnaká. Keďže ostatná útvary majú viac zmien smeru ako obdĺžnik, je pravdepodobné, že trasa 2D KF bude odkláňať zo správnej trasy.



Obr. 4.9: Šesťuholník s hodnotami 0,001 na diagonále Q



Obr. 4.10 Kružnica s hodnotami 0,001 na diagonále Q



Obr. 4.11: Sínusoida s hodnotami 0,001 na diagonále Q

Predpoklad odklonenia sa trajektórie z ideálnej trasy sa potvrdil. Aj keď pri kružnici a sínusoide 2D KF zachováva ideálnu trajektóriu, je výrazne mimo správnej trasy. Tu je vidieť, že hodnoty matice Q je treba voliť opatrne.

Pre vygenerované stratégie som hľadal hodnoty matice Q také, aby bola pri všetkých tvaroch dráha 2D Kalmnanovho filtra lepšia ako dráha ovplyvnená Gaussovým šumom.

Okrem vizuálnej kontroly som porovnával súradnice ovplyvnené šumom a súradnice spočítané 2D KF aj porovnaním ich priemernej vzdialenosti od pôvodne vygenerovaných súradníc. Postup porovnania je nasledovný:

- Postupne v poradí v akom majú súradnice nasledovať spočítam vzdialenosť medzi pôvodnou súradnicou a súradnicou ovplyvnenou šumom (resp. spočítanou 2D KF);
- Následne sa z rozdielov spočíta aritmetický priemer.

Výsledok tohto porovnania, pre najlepšie parametre Kalmanovho filtra tohto testu, je zobrazený v tabuľke 4.5. Hodnoty vzdialeností sú v pixeloch.

Obdĺžnik	
Priemerná vzdialenosť dát obsahujúcich šum	18,781
Priemerná vzdialenosť dát spočítaných KF	12,095
Kruh	
Priemerná vzdialenosť dát obsahujúcich šum	17,520
Priemerná vzdialenosť dát spočítaných KF	12,728
Šesťuholník	
Priemerná vzdialenosť dát obsahujúcich šum	17,841
Priemerná vzdialenosť dát spočítaných KF	12,287
Sínusoida	
Priemerná vzdialenosť dát obsahujúcich šum	17,983
Priemerná vzdialenosť dát spočítaných KF	14,793

Tab. 4.5: Porovnanie úspešnosti 2D KF

Z tabuľky 4.5 je vidieť, že keď sa nastavia parametre tak, aby čo najlepšie zodpovedali všetkým trajektóriám, tak nie je zlepšenie také výrazné (napr. pri sínusoide). Vo všeobecnosti je ale ukázané, že Kalmanov filter môže dobre pomôcť.

4.4 Kalmanov filter pre synchronizáciu

Problému synchronizácie času som sa venoval v kapitole 3. Keďže sa pri synchronizácií používa minulá hodnota koeficientu a chcelo by to hodnotu aktuálnu. Koeficient sa spočíta ako rozdiel časov príchodov dvoch po sebe idúcich synchronizačných správ. Táto hodnota koeficientu je presná pre minulý časový interval, ale v okamihu získania (a to znamená aj v okamihu použitia) už beží ďalší časový interval. Riešením je počítať spätne (poloha tagu sa určí až v momente, kedy bude možné určiť koeficient pre danú periódu). Nevýhodou tohto riešenia je latencia. Interval má dve sekundy a tieto dve sekundy by sa čakalo na výpočet koeficientu. Pre zobrazenie v reálnom čase by to bolo príliš veľké oneskorenie.

Preto by bolo vhodné nájsť spôsob, ako predpovedať hodnotu koeficientu na základe predchádzajúcich koeficientov. Jednou z možností je lineárna aproximácia koeficientov. Alebo používanie starej hodnoty (z predchádzajúceho intervalu).

Pre predpovedanie hodnoty bol vybraný Kalmanov filter. Tento problém nie je až tak rozšírený ako predpovedanie dráhy objektu. Preto v tomto prípade nejde len o to dobre nastaviť Kalmanov filter, ale zistiť či je vôbec vhodný pre túto úlohu. Z rôznych zdrojov, ktoré som našiel vyzeral najsľubnejšie nasledovný [15]. Používal jednoduchý model popísaný stavovou prechodovou maticou A:

$$A = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix}$$

kde dt je časový interval medzi prijatiami signálu. Dvojdimenzionálny stavový vektor x má tvar:

$$x = \begin{Bmatrix} timeOfArrival \\ clockSkew \end{Bmatrix}$$

kde $timeOfArrival$ je čas príchodu signálu na danú kotvu. V tomto zdroji boli uvedené aj hodnoty matíc Q a R. Pre hodnoty šumu vzniknutého pri meraní (matica R) udávajú hodnotu 3×10^{-20}. Pre hodnoty procesného šumu (matica Q) zase udávajú hodnotu 5×10^{-20}. Keďže nás zaujíma hodnota $time\ of\ arrival$ tak matica H bude vyzerať takto:

$$H = [1 \quad 0]$$

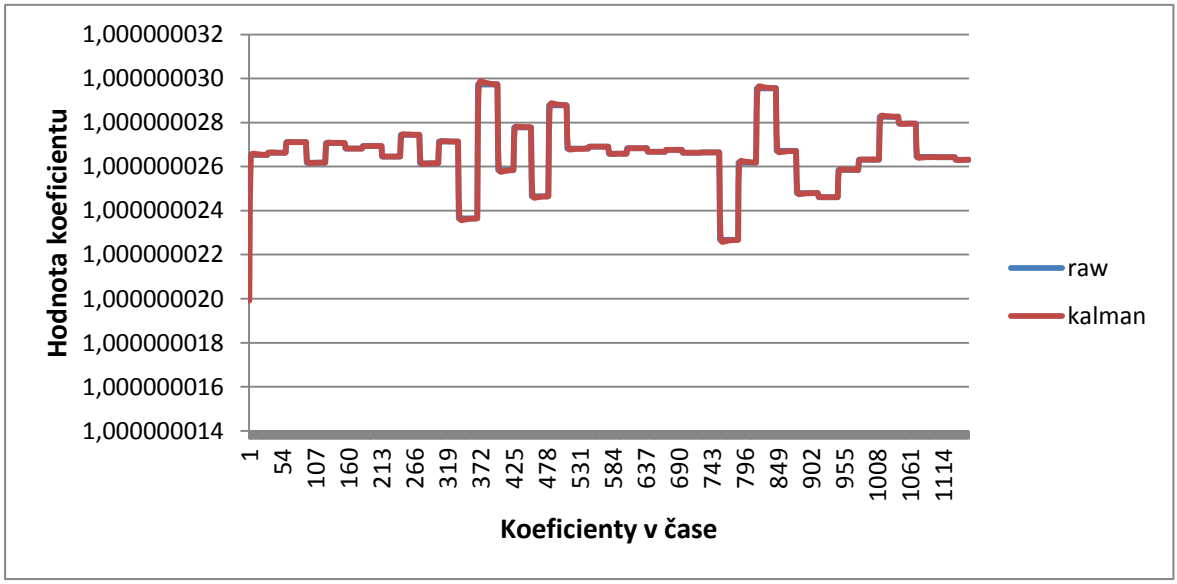
4.4.1 Testovanie

Kalmanov filter pre synchronizáciu (ďalej len KFS) som nepoužil priamo na časy príchodu signálu na kotvu (TOA) ale až na výsledný koeficient. Použitie na časy príchodu signálu na kotvy prináša v súčasnosti príliš veľa problémov. Medzi dané problémy patria napr. straty paketov. Keďže je koeficient spočítaný z časov príchodov signálov na kotvy, tak je možné nahradiť $timeOfArrival$ vektoru x hodnotou koeficientu. V mojom prípade bude teda stavový vektor x vyzerať takto:

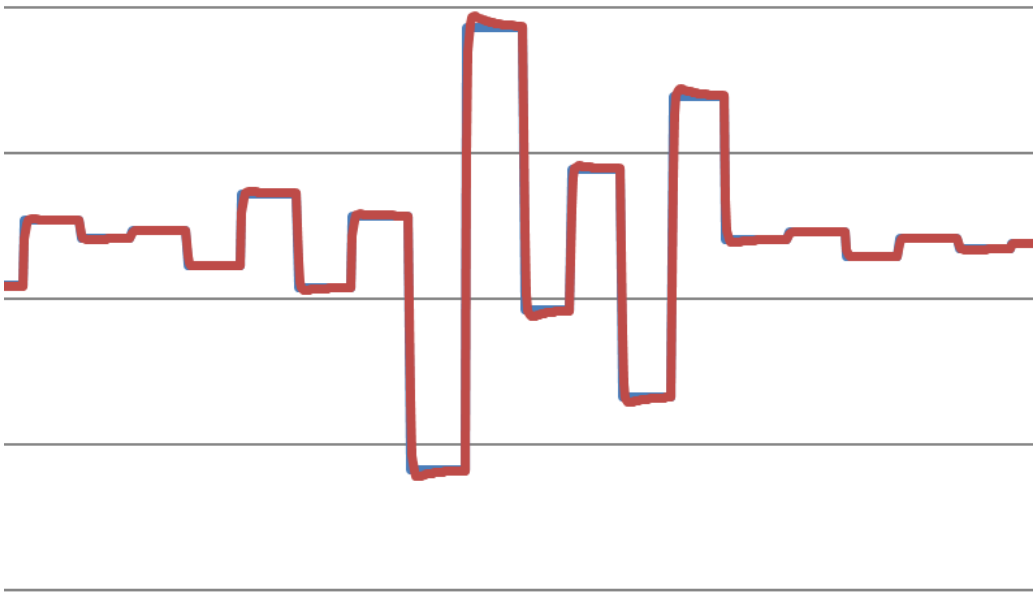
$$x = \begin{Bmatrix} coefficient \\ clockSkew \end{Bmatrix}$$

To znamená, že nebudem Kalmanov filter používať na predpovedanie času príchodu signálu, ale budem predpovedať priamo koeficient nadchádzajúceho intervalu.

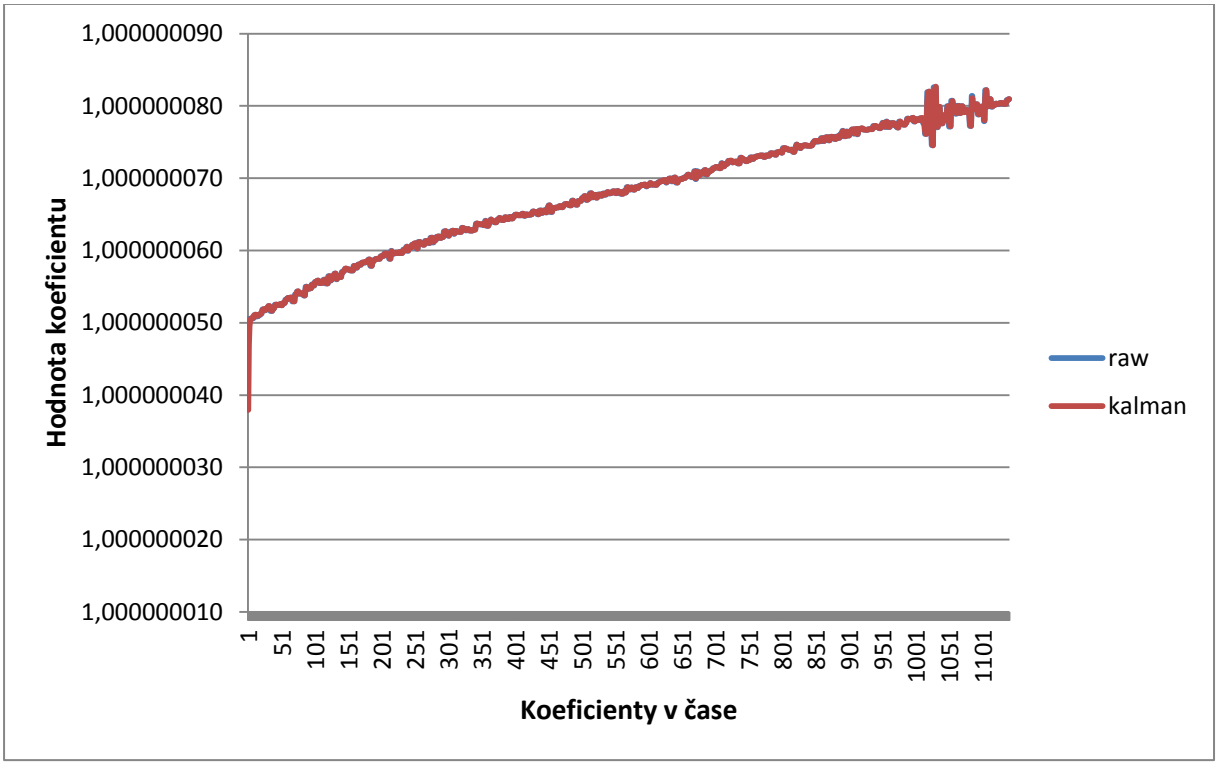
Na sériu dát (hodnôt koeficientov) som aplikoval KFS. Použil som model popísaný v predchádzajúcej kapitole. Namerané hodnoty a hodnoty získané pomocou KFS som porovnal pomocou grafov. Na nasledujúcich grafoch (obr. 4.12, obr. 4.13 a obr. 4.14) je vidieť, že KFS takmer presne kopíruje namerané hodnoty. Keďže čas sa až tak skokovo nemení, lepšie by bolo, keby KFS vypočítal hodnoty, ktoré by pekne (plynulo) „preložili“ namerané hodnoty. Na obr. 4.14 je vidieť pri konci (približne posledných 200 hodnôt), že signál nebol dobrý (v skutočnosti bol rušený pomocou alumíniovej fólie, ktorou zatieňoval priestor pred kotvou, aby prišlo k rušeniu signálu). Práve v takých prípadoch by mal KFS odhadnúť lepšiu hodnotu a nie kopírovať šum.



Obr. 4.12: Rušený signál – KFS kopíruje nerané hodnoty.



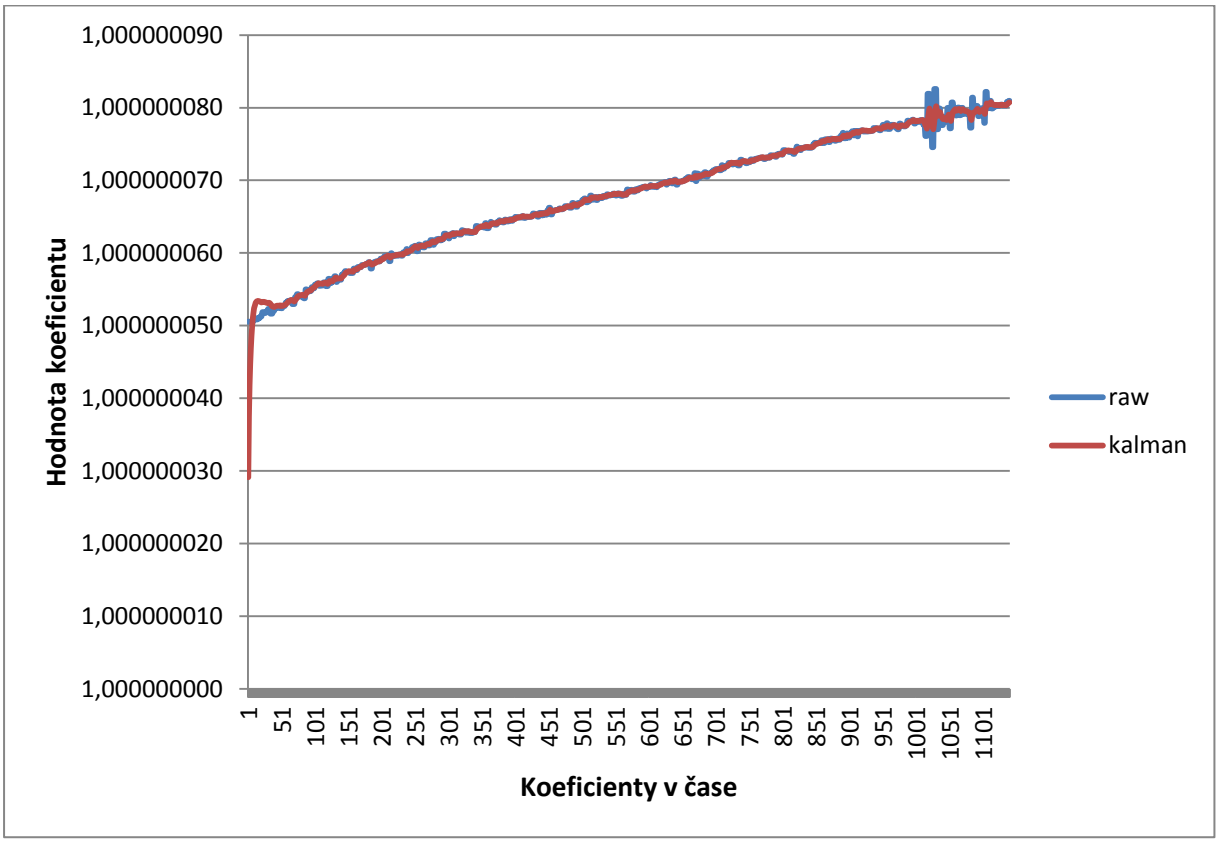
Obr. 4.13: Výrez z grafu na obr. 12



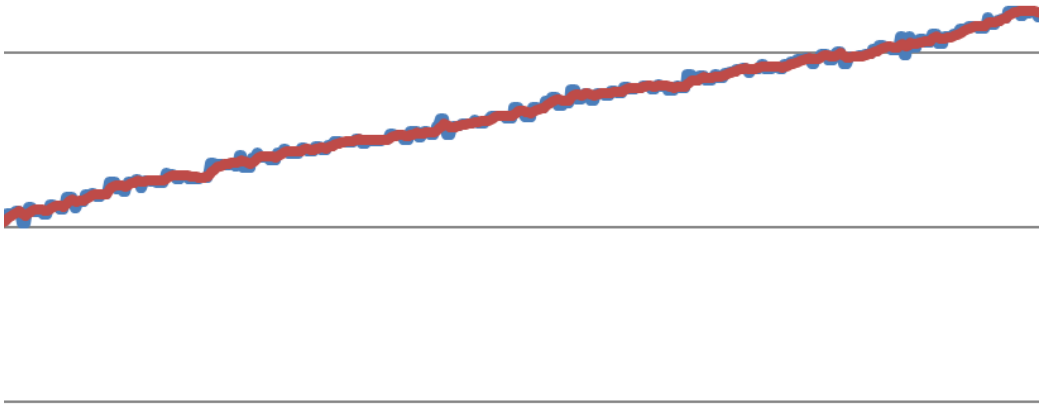
Obr. 4.14: Rušený signál na konci meraného intervalu – KFS kopíruje merané hodnoty.

Vylepšenie hodnôt KFS som sa pokúsil získať znižovaním hodnôt v matici Q. Podľa informácií z [15] bola hodnota na diagonále matice Q rovná 5×10^{-20}. Rozhodol som sa pre postupné znižovanie exponentu o hodnotu dva, t.j. exponenty -22, -24, atď. Výsledky som porovnával na grafoch. Aj keď som porovnával viacero dátových súborov, vývoj zobrazím len na jednom grafe.

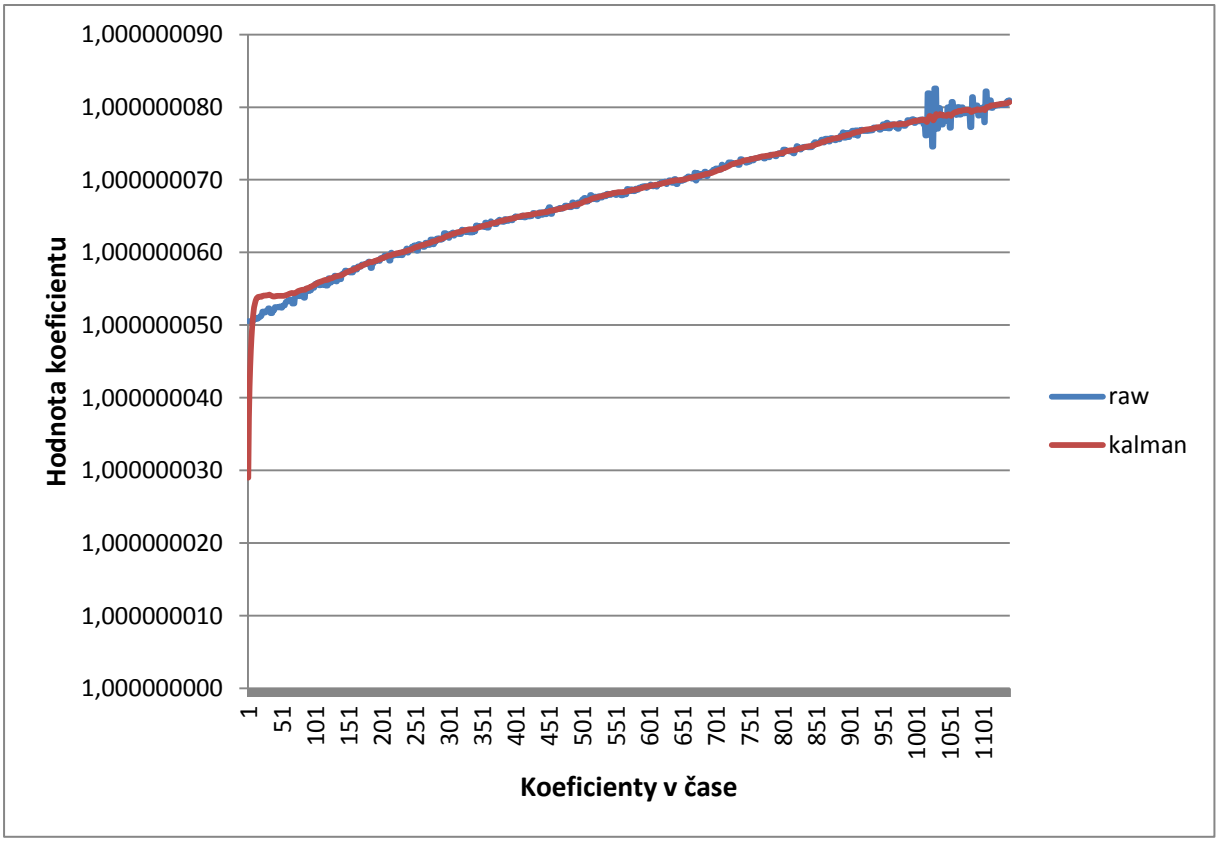
Už pri prvom zmenšení exponentu je vidieť mierne vyhladenie hodnôt KFS. Na obrázkoch obr. 4.15 až obr. 4.20 sú grafy (príp. výrezy z grafov), ktoré zobrazujú postupné znižovanie hodnôt exponentu.



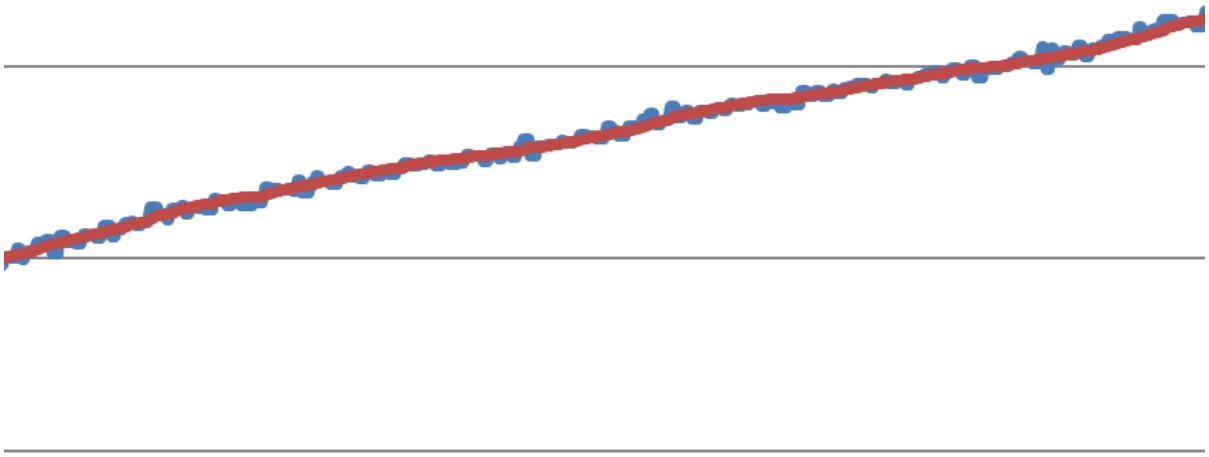
Obr. 4.15: Hodnota exponentu -22



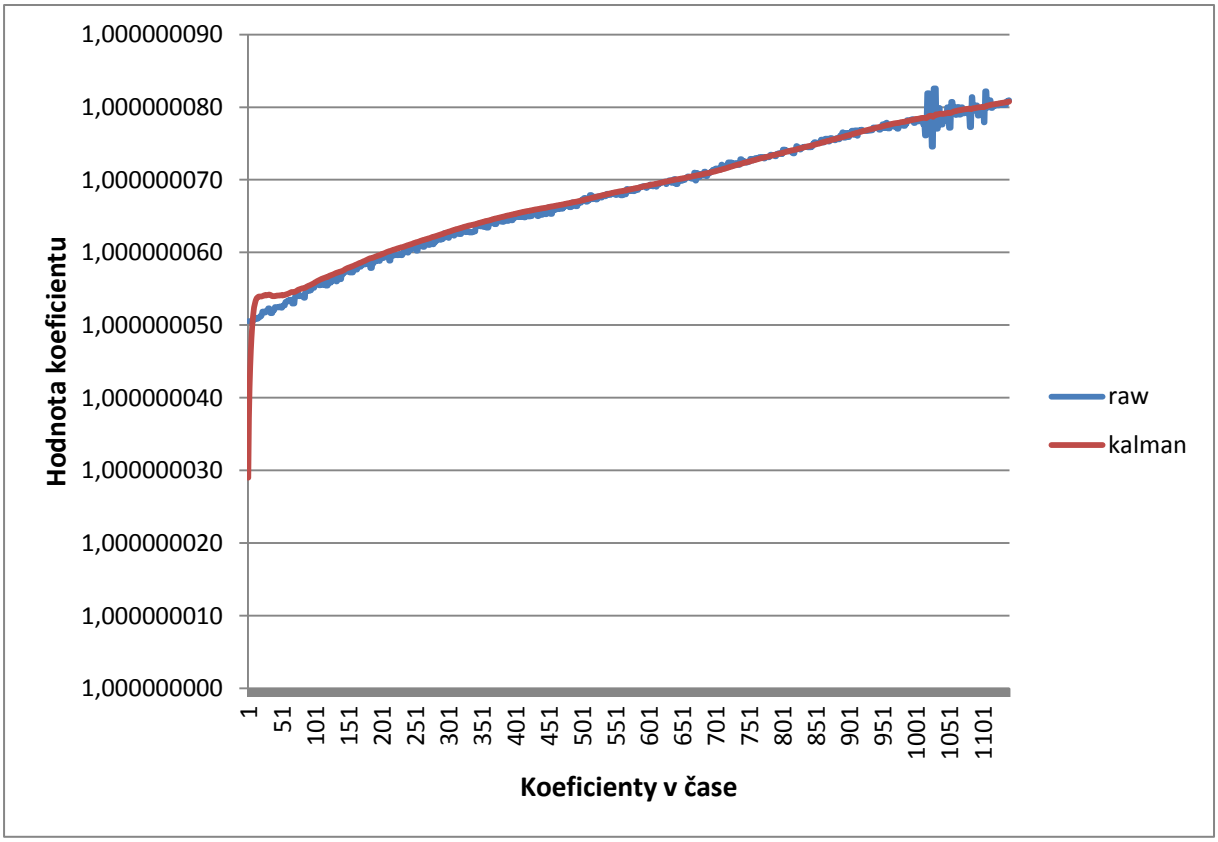
Obr. 4.16: Výrez z grafu na obr. 4.15



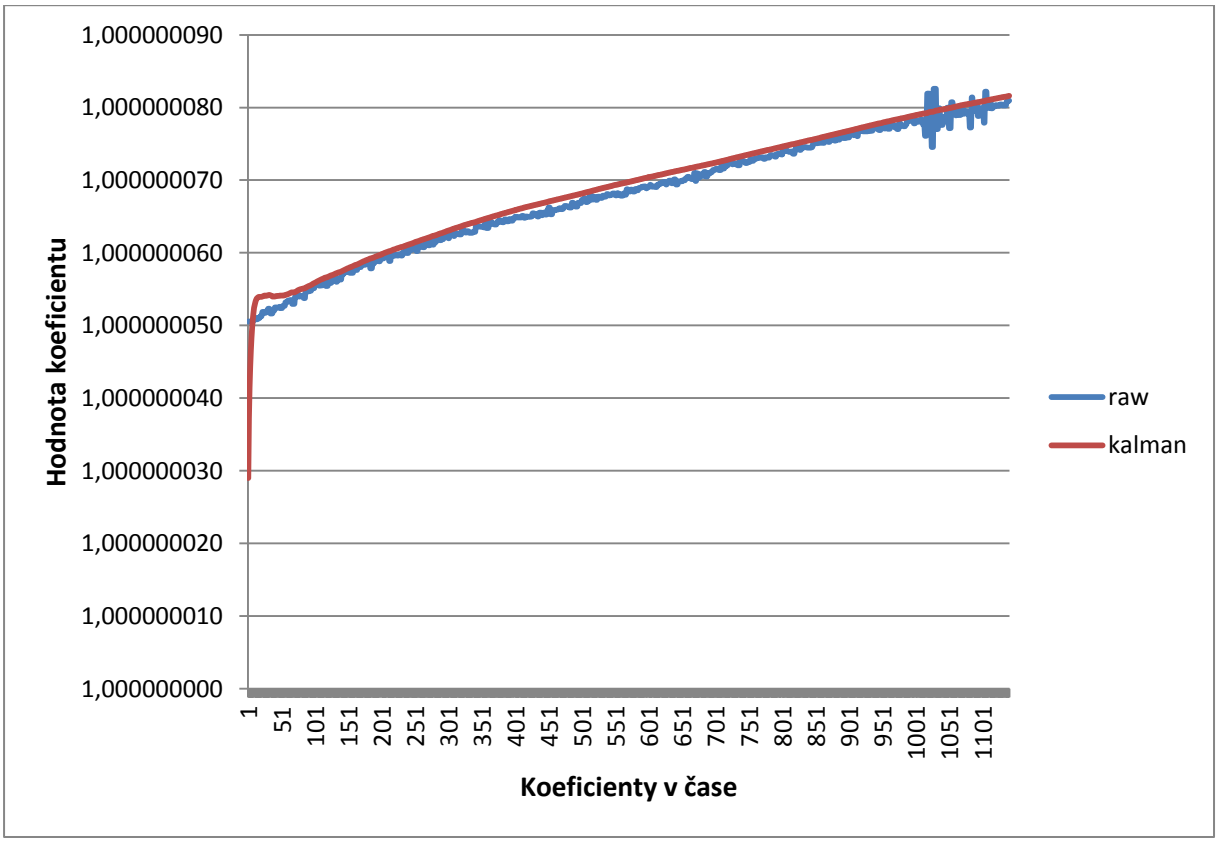
Obr. 4.17: Hodnota exponentu -24



Obr. 4.18: Výrez z grafu na obr. 4.17



Obr. 4.19: Hodnota exponentu -26



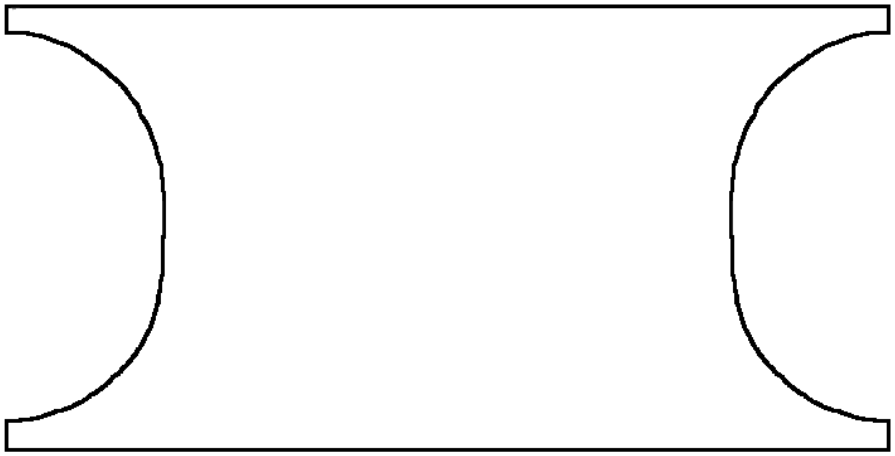
Obr. 4.20: Hodnota exponentu -28

Na poslednom obrázku zo série grafov t.j. obr. 4.20 je vidieť, že KFS už začína počítať hodnoty, ktoré už vystupujú mimo nameraných hodnôt.

Na základe grafov bola určená hodnota exponentu -22 ako najlepšia. Aj napriek tomu, že pri tomto exponente nie je graf hodnôt spočítaných KFS taký plynulý ako pri nižších exponentoch, je tam vidieť lepší odhad koeficientu. Nemenej dôležité je aj to, ako dlho trvá spočítaným hodnotám, kým sa priblížia k reálnym. Je to vidieť na obrázkoch 4.15, 4.17, 4.19 a 4.20, keď na začiatku merania je červený graf nad modrým. Pre relatívne rýchly návrat k reálnym hodnotám bol vybraný exponent -22. A teda najlepšie hodnoty na diagonále matice Q sú 5×10^{-22}.

4.5 Hľadanie parametrov Kalmanovho filtra na reálnych dátach

Po implementácií a overení (uvedomení si) funkčnosti Kalmanovho filtra som začal hľadať optimálne hodnoty matíc Q a R pre reálne dáta zo systému SEWIO. Reálne dáta boli získané na hádzanárskom ihrisku. Trajektória, na ktorej boli dáta namerané je zobrazená na obr. 4.21.



Obr. 4.21: Trajektória pohybu pri meraní reálnych testovacích dát

Keďže tag ohlasuje svoju polohu každých 100 ms, tak hodnotu Δt som nastavil na 0,1. Ostatné premenné (resp. matice) som už musel iba odhadnúť. Keďže nie je žiaden presný návod ako postupovať, prípadne s akými hodnotami začať (každý systém je v realite iný), tak som si počiatočné hodnoty zvolil ľubovoľne:

$$Q = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}$$

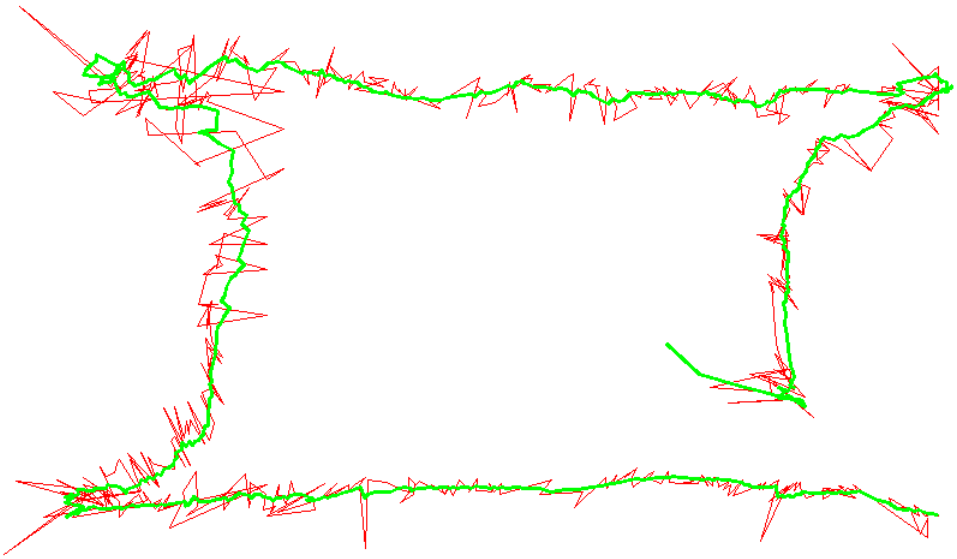
$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

V tomto prípade som už nemal k dispozícií ideálnu trasu tagu, takže som sa snažil na základe grafických výstupov reálnych dát a spočítaných dát určiť, kedy je vypočítaná trasa použiteľná. Na šiestich sadách dát som testoval rôzne hodnoty matíc Q a R, pričom som dodržiaval princíp popísaný v kapitole 4.1 (t.j. hodnoty na diagonále matice Q zmenšovať a hodnoty na diagonále matice R zväčšovať).

Týmto spôsobom som sa dopracoval k takýmto hodnotám:

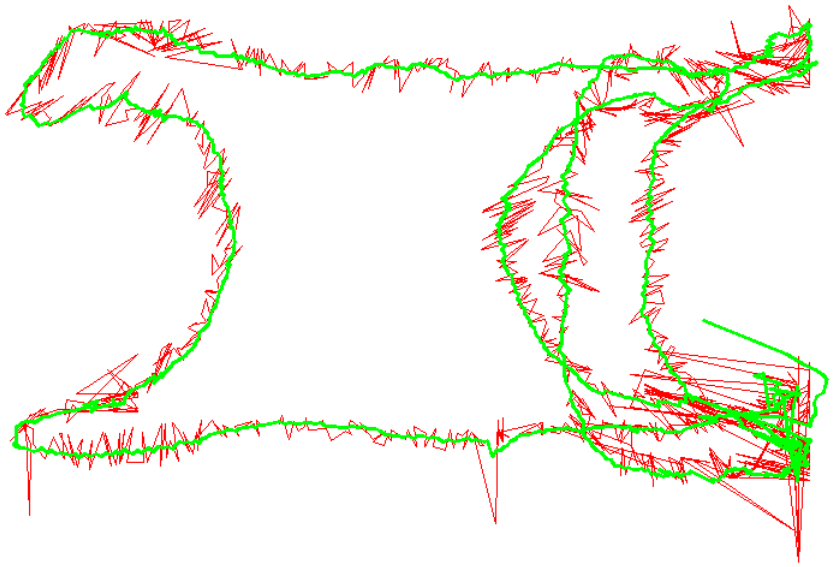
$$Q = \begin{bmatrix} 0.008 & 0 & 0 & 0 \\ 0 & 0.008 & 0 & 0 \\ 0 & 0 & 0.008 & 0 \\ 0 & 0 & 0 & 0.008 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$$



Obr. 4.22: Aplikácia Kalmanovho filtra na reálne dáta

Na obr. 4.22 je vidieť spočítaná trasa pomocou Kalmanovho filtra (zeleným) a pôvodné dáta (červenou). Je dosť zreteľne ukázané, ako môžu rôzne negatívne vplyvy pôsobiť na výslednú polohu. Červená trasa je viac než chaotická. Meranie tejto trasy nezačalo na tom istom mieste ako skončilo. Z toho dôvodu vznikla na obrázku medzera v trajektórií. Na obr. 4.23 je taktiež zobrazený výsledok Kalnamovho filtra len pre iné vstupné dáta. Okrem trajektórie z obr. 4.21 bola trasa doplnená pohybom v pravej časti.



Obr. 4.23: Aplikácia Kalmanovho filtra na reálne dáta

4.6 Kalmanov filter pre trojrozmerné pozičné dáta

Keďže pozícia sa dá určovať aj v trojrozmernom priestore, skúsil som nájsť a vyskúšať model Kalmanovho filtra pre trojrozmerné pozičné dáta. Rozhodol som sa pre nasledujúci model. Stavový vektor x má tvar:

$$x = \begin{Bmatrix} position_x \\ position_y \\ position_z \\ velocity_x \\ velocity_y \\ velocity_z \\ acceleration_x \\ acceleration_y \\ acceleration_z \end{Bmatrix} \tag{4.9}$$

Rovnako ako pre dvojrozmerné dáta, $velocity_x$ a $acceleration_x$ (takisto aj pre zvyšné osi) znamenajú rýchlosť a akceleráciu v smere osi x (resp. danej osi). Ako je možné vidieť z rovnice 4.9, v tomto modeli sa nachádza aj možnosť merania akcelerácie. Matica A je definovaná nasledovne:

$$A = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{(\Delta t)^2}{2} & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{(\Delta t)^2}{2} & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{(\Delta t)^2}{2} \\ 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Pre odhad polohy vyzerá matica H takto:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Po implementácií daného modelu som použil podobný postup ako pri testovaní 2D KF. Vygeneroval som si súradnice (tento krát tri sady – x-ové, y-ové a z-ové) a na tie použil rovnaký šum ako predtým. Následne som chybové súradnice použil ako vstup Kalmanovho filtra.

Súradnice vypočítané Kalmanovým filtrom som porovnal so súradnicami obsahujúcimi šum. Použil som priemernú vzdialenosť oboch typov súradníc od pôvodne vygenerovaných súradníc. Výsledky je možné vidieť v tabuľke 4.6, hodnoty vzdialeností sú znova uvádzané v pixeloch.

Priemerná vzdialenosť dát obsahujúcich šum	22,833
Priemerná vzdialenosť dát spočítaných KF	15,718

Tab. 4.6: Porovnanie úspešnosti Kalmanovho filtra pre trojrozmerné dáta

Z tabuľky je vidieť, že Kalmanov filter priniesol zlepšenie približne o tretinu. Čo zodpovedá testovaniu na obdĺžnikových dátach v predchádzajúcom prípade 2D KF ako je vidieť v tabuľke 4.5. Aj v tomto prípade som vychádzal z tvaru, ktorý mal podstavu obdĺžnika.

5 Implementácia

Implementačná časť mojej diplomovej práce sa dá rozdeliť na tri časti:

- Implementácia Kalmanovho filtra;
- Skripty na spracovanie a zobrazenie výsledkov Kalmanovho filtra;
- Grafická aplikácia porovnávajúca vhodnosť masterov.

V tabuľke 5.1 sú vypísané použité implementačné nástroje.

Operačné systémy		Ubuntu 14.04
		Windows 7 Profesional SP1
Imlementačné jazyky		C
		Python 2.7
Knižnice	C	Meschach
	Python	Opencv
		Numpy
		Math
		PIL
		Tkinter

Tab. 5.1: Použité implementačné nástroje

5.1 Implementácia Kalmanovho filtra

Kalmanov filter je implementovaný ako knižnica v jazyku C. Ku knižnici patria aj adresáre s maticami Kalmanovho filtra. Ako implementačné prostredie slúžil virtuálny operačný systém Ubuntu 14.04. Jazyk C bol určený kvôli rýchlosti výpočtu. Pre maticové výpočty bola zvolená knižnica `Meschach`, ktorá implementuje rôzne výpočty nad vektormi a maticami v jazyku C.

Funkcionalita je pomerne jednoduchá a pozostáva z týchto krokov:

- Inicializácia matíc – Načítanie hodnôt matíc do premenných. Matice popisujúce systémový model a šum v systéme sú uložené v textových súboroch. Zároveň je medzi nimi aj inicializovaný stavový vektor x a kovariačná chyba P;
- Výpočet (odhad) hľadanej hodnoty – V tomto kroku sa počíta odhad stavovej premennej x. Na vstupe sú merané dáta a výstupom je spomínaný odhad;
- Dealokácia matíc.

5.2 Spracovanie a zobrazenie výsledkov Kalmanovho filtra

Pre grafické zobrazenie trajektórií KF som sa rozhodol použiť knižnicu Opencv, ktorá je vhodná na spracovanie obrázkov. Ja som si ju vybral pre jednoduchú tvorbu obrázkov. Každý obrázok vytváraný

ako matica farebných bodov (čiže trojicu (B,G,R)). V Opencv je treba dať pozor na obrátené poradie farieb v štandardnom modely RGB – používa sa model BGR.

Pre implementáciu testovania som si vybral jazyk Python (verziu 2.7) pod operačným systémom Windows 7. Testovanie pozostávalo z viacerých krokov. Prvým bolo overenie funkčnosti (prípadne sa dá povedať funkcionality) KF. Pre tento účel som si vytvoril štyri skripty, ktoré vygenerovali testovacie trajektórie pre KF:

- `circle.py`;
- `hexagon.py`;
- `rectangle.py`;
- `sine.py`.

Výsledkom daných skriptov boli textové súbory so súradnicami vytvárajúcimi jednotlivé geometrické obrazce a súbory, ktoré obsahovali chybové dáta, t.j. súradnice, ku ktorým sa pričítali náhodné hodnoty z Gaussovho šumu.

Ďalšie skripty, ktoré som vytvoril posudzovali úspešnosť KF. Prvý (`write_tracks.py`) vytvorí obrázky a videá s všetkými troma trajektóriami (vzorovou, chybovou, spočítanou KF). Na obrázkoch je možné porovnať trajektórie vizuálne. Na videách je vidieť postupnú tvorbu jednotlivých trajektórií (niečo ako rozhodovanie KF v reálnom čase). Druhý skript na porovnanie úspešnosti (`stats.py`) zisťuje priemernú vzdialenosť chybových dát a dát spočítaných pomocou KF od vzorových dát. Na zistenie vzdialenosti (*angl. distance*) súradníc (chybový, resp. odhadnutých KF) od vzorových súradníc používa štandardnú euklidovskú metriku:

$$distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

kde x_i a y_i sú súradnice bodu v 2D priestore. Výstupom skriptu je textový súbor porovnávajúci maximálne a minimálne hodnoty dát a priemernú chybu dát. Súbor obsahuje hodnotu priemernej vzdialenosti (*angl. average distance*) medzi vygenerovanými dátami a dátami ovplyvnenými šumom (resp. dátami odhadnutými KF). Hodnota vzdialenosti medzi generovanými a šumom ovplyvnenými dátami je označená ako `orig - noise`. Hodnota vzdialenosti medzi generovanými a KF odhadnutými dátami je označená ako `orig - kalman`. Pre lepšiu predstavu o dátach sú pridané aj informácie o maximálnej a minimálnej vzdialenosti medzi danými dátami. Tieto hodnoty sú pre každý jeden generovaný objekt: obdĺžnik (`RECTANGLE`), kružnicu (`CIRCLE`), šesťuholník (`HEXAGON`) a sínusoidu (`SINE`). Príklad výstupu:

```
RECTANGLE:
Average distance  orig - noise:   18.7813625407
Average distance  orig - kalman:  10.0848957884

Maximal distance orig - noise:    55.2177507691
Minimal distance orig - noise:    1.0
Maximal distance orig - kalman:   31.5973665
Minimal distance orig - kalman:   0.649884305342
CIRCLE:
Average distance  orig - noise:   17.5197111684
Average distance  orig - kalman:  23.7133047383

Maximal distance orig - noise:    39.3573373083
Minimal distance orig - noise:    2.0
Maximal distance orig - kalman:   41.0017976714
```

```
Minimal distance orig - kalman:   3.70030088857
HEXAGON:
Average distance  orig - noise:   17.841197399
Average distance  orig - kalman:  11.7750021238

Maximal distance orig - noise:    48.0832611207
Minimal distance orig - noise:    0.0
Maximal distance orig - kalman:   23.7318865207
Minimal distance orig - kalman:   0.729146604499
SINE:
Average distance  orig - noise:   17.9833026653
Average distance  orig - kalman:  32.0012781547

Maximal distance orig - noise:    54.7083174664
Minimal distance orig - noise:    1.0
Maximal distance orig - kalman:   66.8315012419
Minimal distance orig - kalman:   4.32477729157
```

Skripty sú spúšťané z príkazového riadku a v úvodných riadkoch sú hodnoty, ktoré možno meniť podľa potreby. Kvôli množstvu testov som menil hlavne adresáre obsahujúce dáta, hrúbku čiar a počet pixelov medzi bodmi.

5.3 Aplikácia porovnávajúca vhodnosť masterov

Treťou časťou je grafická aplikácia, taktiež implementovaná v jazyku Python 2.7, ktorá zistí ideálneho mastera pre dané rozloženie systému (polohu kotiev). Ideou je situácia, kedy je tag staticky umiestnený v určitom bode. Tento bod je známy. Systém sa spustí niekoľko krát, vždy s iným masterom. Pri každom spustení sa merajú hodnoty pozície tagu. Aplikácia na základe rozptylu určí najlepšieho mastera. Taktiež zobrazí kotvy, tag a namerané hodnoty.

Určenie najlepšieho mastera pred konkrétnu situáciu je problém, ktorý sa dá vyriešiť len empirickým meraním. Príčiny vhodnosti istej kotvy oproti inej v danom systéme môžu byť rôzne:

- Stabilnejší oscilátor – master so stabilnejším oscilátorom bude vykazovať lepšie výsledky;
- Menej odrazov pri šírení synchronizačných správ – nevhodne zvolený master, ktorý má zakrytý výhľad na ostatné kotvy. V takom prípade dochádza k rôznym odrazom signálu a teda aj predĺženiu trajektórie signálu, čo spôsobuje nepresnosti v systéme.

Do aplikácie sa načítajú textové súbory s pozíciami kotiev a nameranými hodnotami tagu. V súbore je tiež určené, ktorá kotva bola masterom. Tag sa zadáva manuálne, aby bolo možné porovnávať jednotlivé merania. Súbor môže vyzerať nasledovne:

```
ANCHORS
36.37547    16.49798    D88039629337
51.00626    16.68123    D880396232F7
51.025020    20.337740     D8803961E7D0
MASTER       D8803961E7D0
TAG
22.504979    16.859469
22.462768    16.864713
22.645230    16.868343
```

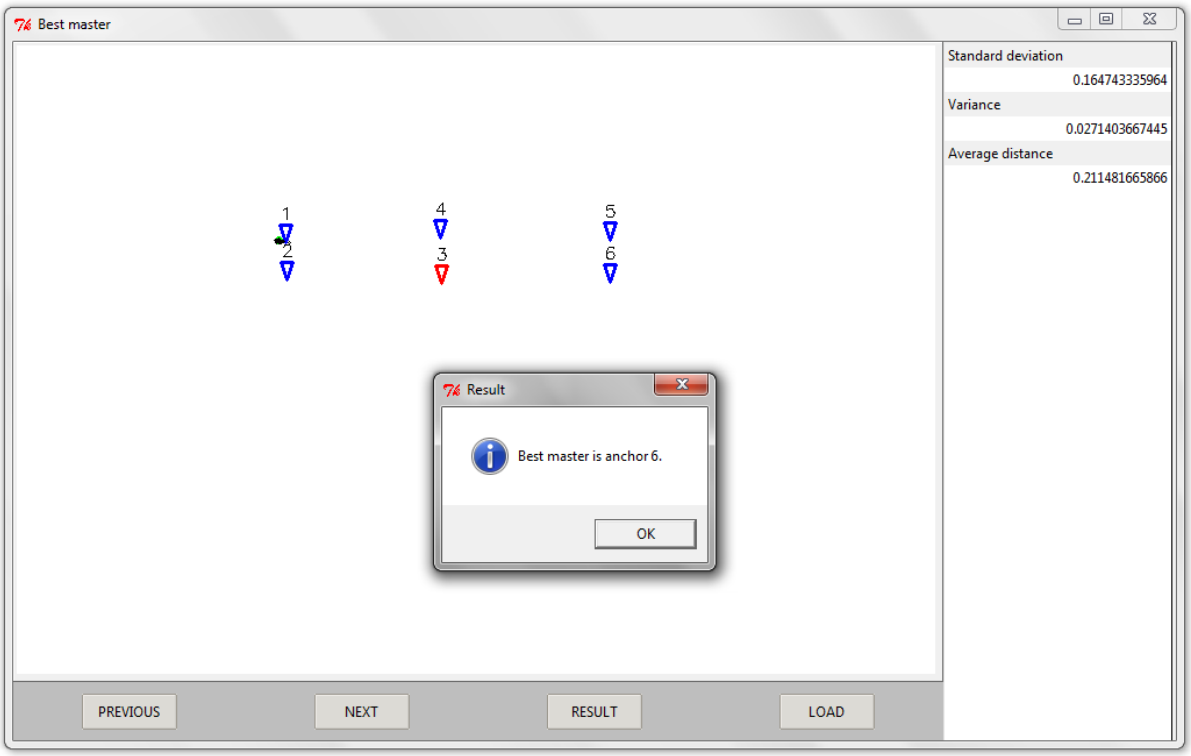
```
22.339134     16.761855
22.660603     16.876832
```

Na začiatku súboru sú kotvy označené kľúčovým slovom `ANCHORS`. Sú definované svojou pozíciou (súradnice *x* a *y*) v priestore (prvé dve hodnoty) a adresou kotvy (tretia hodnota). Nasleduje kľúčové slovo `MASTER`, vedľa ktorého je adresa mastera. To určuje, ktorá kotva v systéme bola masterom. Za definovaním mastera je kľúčové slovo T. Označuje namerané hodnory pozície tagu (súradnice *x* a *y*).

Práca s aplikáciou zahŕňa:

- Načítanie súborov s dátami (pri prvom načítaní sa zadáva hodnota tagu). Po načítaní sa schéma systému kotiev vykreslí;
- Prepínanie medzi obrázkami načítaných schém;
- Vypísanie najlepšieho mastera.

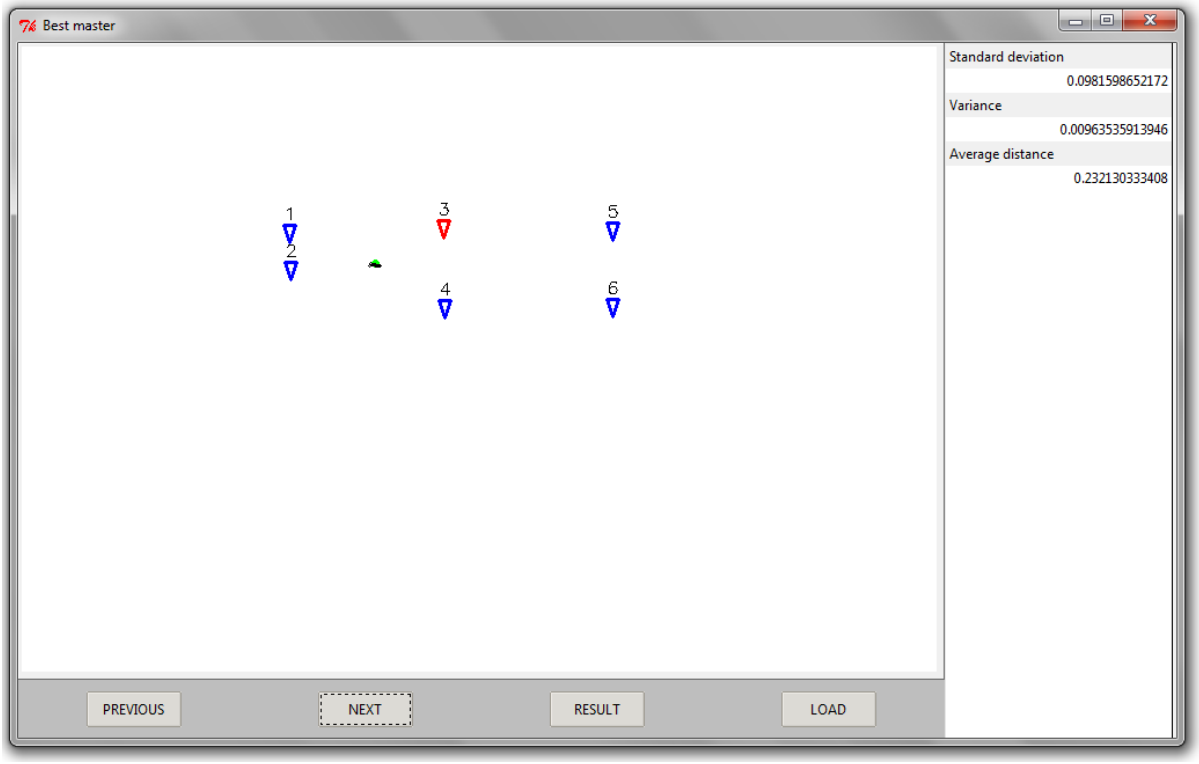
Na obrázku 5.1 je príklad spustenej aplikácie aj s výpisom najlepšieho mastera. V časti na vykreslenie schémy sú kotvy znázornené modrými trojuholníkmi (master červeným), tag zeleným bodom (pre slabú viditeľnosť zobrazený na obr. 5.2) a namerané dáta. V pravej časti sú štatistické údaje a v dolnej tlačidlá na ovládanie.



Obr. 5.1: Aplikácia na porovnanie masterov

Obr. 5.2: Výrez z obr. 5.1

Aplikácia bola použitá pre meranie v miestnosti s pôdorysom v tvare písmena *L*. Bolo použitých šesť kotiev, pričom nie všetky kotvy mali priamu viditeľnosť na ostatné. Rozmiestnenie kotiev je na obr. 5.3. Na danom obrázku je celá testovaná situácia. Tag bol umiestnený do priestoru medzi kotvy ①, ②, ③ a ④. Priamu viditeľnosť na ostatné kotvy mali len kotvy ③ a ⑤.



Obr. 5.3: Rozmiestnenie kotiev pri meraní

Postupne sa za mastera volili kotvy v poradí ich číslovania a získali sa vstupné súbory pre aplikáciu. V aplikácií sa jednotlivé súbory načítali a vyhodnotili. Vyhodnotenie je v tabuľke 5.2. Primárnym hodnotiacim prvkom bol rozptyl vzdialeností od zadaného bodu. Z vyhodnotenia vyplynulo, že najlepším masterom je kotva ③. Kotva ④ dopadla najhoršie, zrejme preto, že okrem zlej viditeľnosti na kotvy ① a ②, nemala ani priamu viditeľnosť na tag.

Kotva	Rozptyl
①	0,0282143142607
②	0,0397127239777
③	0,0096353591394
④	0,1135582510917
⑤	0,0388896714525
⑥	0,1041693986599

Tab. 5.2: Vyhodnotenie merania

Aplikácia by sa dala rozšíriť (vylepšiť) na načítanie dát z jedného súboru. Momentálne treba taký počet súborov, koľko je kotiev (resp. koľko kotiev je testovaných ako master). Zlepšiť by sa mohol aj jednoduchý grafický náhľad na rozostavenie kotiev.

Zaujímavou možnosťou by bolo, keby nebol tag statický, ale pohyblivý. Bolo by možné nielen kontrolovať efektivitu mastera, ale aj zistiť, v ktorých oblastiach sa poloha tagu spočítala horšie (napr. za nejakou prekážkou, rohom, a i.). To by ale potrebovalo mať presnú dráhu tagu, čo sa mnohokrát ťažko získava.

6 Záver

V mojej diplomovej práci som postupne rozobral najznámejšie meracie techniky v bezdrôtových lokalizačných systémoch. Následne som popísal systém spoločnosti SEWIO, pre ktorú robím svoju diplomovú prácu. V popise systému SEWIO som sa zameral na výpočet polohy tagu a synchronizáciu času na kotvách vrátane výpočtu koeficientu, ktorý udáva pomer rýchlosti plynutia času medzi masterom a kotvami.

Dôležitou časťou tejto práce je popis Kalmanovho filtra, ako jednej z možností vylepšenia presnosti interiérovej lokalizácie. Podrobne bol popísaný Kalmanov filter pre pozičné dvojrozmerné dáta a následne aj trojrozmerné dáta. Boli vysvetlené premenné vyskytujúce sa v algoritme Kalmanovho filtra a hlavne vplyv matice Q a R, ktorých určenie má významný dosah na správnu funkcionalitu Kalmanovho filtra. V kapitole 4.3 bol na sérií testov s predpripravenými dátami vysvetlený zmysel matice Q pre Kalmanov filter.

Pre predpovedanie koeficientov v synchronizačných dátach bol nájdený a implementovaný a otestovaný model.

Na testovanie Kalmanovho filtra bola v rámci tejto práce vytvorená C knižnica, ktorá bude zahrnutá do ďalšej vydávanej verzie systému SEWIO. V priebehu vývoja bola využívaná na testovanie vhodných parametrov systému v „offline“ móde. Testy boli opakované na niekoľkých sériách dát a výsledkom testovania boli hľadané parametre systému, ktoré boli použité v reálnom prostredí.

Práca s knižnicou mi umožňovala testovanie a rôzne experimenty nad reálne nameranými hodnotami. Pre reálne použitie bude potrebné v tejto knižnici upraviť niektoré časti. Bude treba zmeniť výstup Kalmanovho filtra, ktorý je aktuálne zapisovaný do súboru. Zmena spočíva v presmerovaní výstupu (aktuálne je to výpis dát do súboru) Kalmanovho algoritmu späť do systému, v ktorom sa dáta ďalej spracujú. Zmena by mohla byť v tom, že funkcia by tieto spočítané hodnoty vracala.

Súčasťou práce bola aj analýza alternatívnych riešení Kalmanovho filtra. Tieto alternatívy Kalmanovho filtra sa nepodarilo úspešne implementovať a v budúcom vývoji by sa mohli vylepšiť. Takýmto experimentom bol rozšírený Kalmanov filter (*angl. Extended Kalman filter*), ktorý považujem za zaujímavé rozšírenie, pretože ide o nelineárnu verziu Kalmanovho filtra. Okrem rozšíreného Kalmanovho filtra existujú aj iné rozšírenia (napr. *Unscented Kalman filter*). Porovnanie ich účinnosti by mohlo byť zaujímavé, ak by sa ukázali ako vhodné pre situácie, v ktorých sa momentálne používa Kalmanov filter v systéme SEWIO.

Literatúra

[1] Software Defined Radio Localization Using 802.11-style Communications [online]. WORCESTER [cit. 2016-01-02]. Dostupné z: https://www.wpi.edu/Pubs/E-project/Available/E-project-042811-163711/unrestricted/NRL_MQP_Final_Report.pdf. Správa. WORCESTER POLYTECHNIC INSTITUTE. Vedúci práce Professor Alexander Wyglinski

[2] GPS Accuracy. GPS.GOV: Official U.S. Government information about the Global Positioning System (GPS) and related topics [online]. 2014 [cit. 2016-01-02]. Dostupné z: http://www.gps.gov/systems/gps/performance/accuracy/

[3] HIGHTOWER, Jeffrey a Gaetano BORRIELLO. Location Sensing Techniques [online]. Seattle, 2001 [cit. 2016-01-02]. Dostupné z: http://www.hightowerweb.org/pubs/hightower2001techniques/hightower2001techniques.pdf. Technická správa. University of Washington.

[4] Survey of Wireless Indoor Positioning Techniques and Systems. In: Survey of Wireless Indoor Positioning Techniques and Systems [online]. 2007 [cit. 2016-01-02]. Dostupné z: http://www.pitt.edu/~dtipper/2011/Survey1.pdf

[5] KAUNE, Regina. Accuracy Studies for TDOA and TOA Localization [online]. Wachtberg [cit. 2016-01-02]. Dostupné z: http://fusion.isif.org/proceedings/fusion12CD/html/pdf/056_271.pdf. Technická správa. University of Bonn.

[6] SHIN, Dong-Ho a Tae-Kyung SUNG. Comparisons of error characteristics between TOA and TDOA positioning. IEEE Transactions on Aerospace and Electronic Systems [online]. 2002, vol. 38, issue 1, pp. 307-311 [cit. 2016-01-02]. DOI: 10.1109/7.993253. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=993253

[7] KOČAN, Tomáš. Lokalizační služby ve webových aplikacích. Brno, 2015. Diplomová práca. Vysoké učení technické. Vedoucí práce Veselý Vladimír, Ing.

[8] Kalman filter. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2010-2015 [cit. 2016-01-02]. Dostupné z: https://en.wikipedia.org/wiki/Kalman_filter

[9] KIM, Phil. Kalman filter for beginners: with MATLAB examples. Překlad Lynn Huh. Charleston: CreateSpace, 2011, xiv, 231 s. ISBN 978-1-463648350.

[10] The Technologies behind a Context-Aware Mobility Solution. Cisco Systems, Inc [online]. [cit. 2016-01-02]. Dostupné z: http://www.cisco.com/c/en/us/solutions/collateral/borderless-networks/context-aware-mobility-solution/white_paper_c11-476796.html

[11] Hyperbola. Matematika.cz: tady to pochopíš [online]. 2014 [cit. 2016-01-02]. Dostupné z: http://www.matematika.cz/hyperbola

[12] Dilution of precision (GPS). In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-01-10]. Dostupné z: https://en.wikipedia.org/wiki/Dilution_of_precision_(GPS)

[13] SCHAFFER, Bernhard, Gerrit KALVERKAMP a Erwin BIEBL. A 2.4 GHz high precision local positioning system based on cooperative roundtrip time of flight ranging. In: GeMiC 2014: German Microwave Conference, March, 10 - 12, 2014, Aachen, Germany [online]. Offenbach: VDE-Verl., 2014 [cit. 2016-01-14]. ISBN 3800735857. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6775169

[14] Continuous Random Variables, 2012. 2012 Book Archive [online], 2012. [cit. 2016-01-02]. Dostupné z: http://2012books.lardbucket.org/books/beginning-statistics/s09-continuous-random-variables.html

[15] MCELROY, Ciaran, Dries NEIRYNCK a Michael MCLAUGHLIN. 2014. Comparison of wireless clock synchronization algorithms for indoor location systems. 2014 IEEE International Conference on Communications Workshops (ICC) [online]. IEEE, , 157-162 [cit. 2016-05-16]. DOI: 10.1109/ICCW.2014.6881189. ISBN 978-1-4799-4640-2. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6881189

[16] KANAAN, M. a K. PAHLAVAN. 2004. CN-TOAG: a new algorithm for indoor geolocation. In: 2004 IEEE 15th International Symposium on Personal, Indoor and Mobile Radio Communications (IEEE Cat. No.04TH8754) [online]. Waltham: IEEE, s. 1906-1910 [cit. 2016-04-16]. DOI: 10.1109/PIMRC.2004.1368330. ISBN 0-7803-8523-3. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1368330

[17] MORÉ, Jorge J. 1978. The Levenberg-Marquardt algorithm: Implementation and theory. Numerical Analysis [online]. Springer Berlin Heidelberg, , 105 [cit. 2016-05-09]. DOI: 10.1007/BFb0067700. ISBN 978-3-540-08538-6. Dostupné z: http://www.springerlink.com/index/10.1007/BFb0067700

[18] The Standard Normal Distribution. 2016. OpenStax [online]. [cit. 2016-05-17]. Dostupné z: https://cnx.org/contents/KgL8DwG_@4/The-Standard-Normal-Distributi

Zoznam príloh

Príloha 1 – Obsah DVD

DVD obsahuje tieto adresáre:

- Technická_správa – obsahuje text diplomovej práce vo formáte pdf, vo formáte docx a použité obrázky;
- Knižnica_Kalmanovho_filtra – tu sú zdrojové kódy vytvorené vrámci diplomovej práce ohľadom Kalmanovho filtra a virtuálny OS na otestovanie Kalmanovho filtra;
- Testovanie – obsahuje testovacie skripty a adresáre s dátami (testovacími, vypočítanými Kalmanovým filtrom) ;
- Aplikácia_testovania_najlepsieho_mastera – obsahuje skript danej aplikácie a dáta na overenie funkčnosti.

Stručný popis skriptov, programov a adresárov

Adresár – Testovanie

- circle.py, hexagon.py, rectangle.py, sine.py
    - vygenerujú testovacie dáta (ideálnu trajektóriu a trajektóriu ovplyvnenú šumom);
- read_csv.py
    - spracuje dáta (hodnoty koeficientov), aby som mohol testovať Kalmanov filter pre synchronizáciu;
- stats.py
    - porovná úspešnosť Kalmanovho filtra na základe aritmetického priemeru vzdialeností trajektórie ovplyvnenej šumom a trajektórie spočítanej Kalmanovým filtrom od pôvodnej trajektórie;
- write_tracks.py
    - vytvorí obrázky (príp. video) pre vizuálnu kontrolu účinnosti Kalmanovho filtra pre generované dáta;
- write_tracks-real_data.py
    - vytvorí obrázky (príp. video) pre vizuálnu kontrolu účinnosti Kalmanovho filtra pre reálne dáta;
- _3D
    - obsahuje testovacie dáta pre 3D Kalmanov filter, skript na generovanie 3D dát a skript na otestovanie funkčnosti Kalmanovho filtra (to isté ako popísané stats.py, len pre 3D);
- test_kalman_clock
    - testovacie dáta pre KFS;
    - výsledky testovania zobrazené na grafoch;
- Ostatné adresáre
    - množstvo testovacích dát (reálnych aj generovaných) pre 2D KF.

Adresár – Aplikácia_testovania_najlepsieho_mastera

- best_master.py
    - skript pre aplikáciu porovnania masterov;
- Textové súbory obsahujúce dáta o polohe tagu, kotvách. Sú vstupné údaje pre aplikáciu.

Adresár – Knižnica_Kalmanovho_filtra

- kalman

    - súbory s dátami, ktoré spočítal (resp. spočíta) Kalmanov filter;
- matrix
    - obsahuje adresáre s maticami pre rôzne konfigurácie Kalmanovho filtra;
- noise
    - súbory s dátami ovplyvnenými šumom;
- synch
    - súbory obsahujúce
        1. Vstupné dáta pre KFS;
        2. Dáta spočítané pomocou KFS;
- Kalman.c
    - zdrjový kód knižnice;
- Kalman.h
    - hlavičkový súbor;
- experiments.c
    - zdrojový súbor, ktorým som testoval knižnicu a dáta;
- main.c
    - zdrojový súbor s ukážkovým použitím;
- compile_lib.sh
    - pomocný skript na preklad knižnice;
- compile_program.sh
    - skript na preklad vzorového programu;
- libkalman.a
    - vytvorená knižnica.

Adresár – Technická_správa

- Obsahuje technickú správu vo formátoch pdf a docx. Zároveň obsahuje aj adresár obrázkami použitými v práci.

README

- Popis práce so skriptami a knižnicou.

Príloha 2 – Zoznam obrázkov

Príloha 3 – Zoznam tabuliek