České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů

Diplomová práce

Analyzátor signálu optických vláknových senzorových sítí

Bc. Stanislav Novák

Vedoucí práce: Mařík Radek Ing., CSc

Studijní program: Otevřená Informatika, Magisterské

Obor: Počítačové inženýrství

27. května 2016

Poděkování

Rád bych poděkoval svému vedoucímu práce Ing. Radkovi Maříkovi, CSc. za vedení práce, přínosné rady, podporu a možnost si vyzkoušet výzkum nové technologie v praxi.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 25. 5. 2016 .....

Abstract

Safibra company Ltd. commisioned a requirement to analyse and implement an application for graphic vizualization of measurement data from internal experiments. This thesis solves the problem of data vizualization. In these data are then detected anomalies, which are further analysed by data mining algorithms.

Abstrakt

Společnost Safibra s.r.o. zadala požadavek na analýzu a implementaci aplikace pro grafickou vizualizaci naměřených dat z interně prováděných experimentů. Diplomová práce se zabývá problematikou vizualizace naměřených dat. V těchto datech jsou následně detekovány anomálie, které se dále analyzují dataminingovými algoritmy.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Společnost Safibra s.r.o. se zabývá vývojem optovláknových technologií a senzory. Společnost disponuje vlastním výzkumným a vývojovým oddělením a podílí se na různých výzkumných projektech. Mezi jeden z projektů patří projekt FiberSense. Cílem projektu FiberSense je nalezení liniových a zónových ochran s použitím moderních optických principů a hledání optických technologií, které by umožnily poskytnout inovované metody zajištění výrazně vyšší bezpečnosti kritických infrastruktur oproti současnému stavu při zachování cenové dostupnosti nových zařízení a systémů [4].

Touto problematikou se ve světě zabývají další společnosti pro vývoj optických senzorových sítí. Například společnost Fiber SenSys, Inc., která poskytuje komplexní řešení zabezpečení objektů pro armádu, kde je zapotřebí zabezpečit proti neoprávněnému vniknutí do objektu se skladem zbraní. Dalšími objekty, které jsou potřeba zabezpečit proti např. teroristickým útokům, jsou letiště, přístaviště a jiná zařízení určená k přepravě osob nebo cenného nákladu. Pro zabezpečení těchto objektů používají nejen optické sítě, ale i kombinace optických sítí s mikrovlnými nebo laserovými senzory a kamerovým systémem. [3]

Kapitola 2

Zabezpečovací systémy s optickými senzory

Tato kapitola je věnována hlubšímu představení principů optických senzorových sítí. Budou zde popsány běžné způsoby použití optických vláken pro snímání určité oblasti, která má být zabezpečena. Dále zde bude popsány principy detekce anomálií, které vznikají nějakou externí událostí. Například nedovolené vniknutí do zabezpečené oblasti nebo ve venkovních prostorech může být vyvolána větrem nebo silným deštěm.

2.1 Návrh efektivního zabezpečovacího systému

Návrh účinného a spolehlivého zabezpečení vekovního prostoru je celkem nelehký úkol. Musíme určit rizika specifického místa, které je potřeba zabezpečit, vybrat správné monitorovací mechanismy a předvídat, jaké možnosti má možný narušitel k dispozici. Dalším důležitým bodem je cena provozu zabezpečovacího systému. Jednotlivé druhy instalací zabezpečovacích systémů mají své jedinečné vlastnosti, avšak každý zabezpečovací systém venkovních prostor by měl splňovat následujících pět základních ochranných pravidel:

- `Definice` - správná definice možných narušení
- `Odrazení` - systém zabezpečení by měl odradit útočníka
- `Detekce` - správná detekce, že do objektu vnikl narušitel
- `Zpoždění` - odezva systému, kdy vyhodnotí, že do chráněné zóny vnikl narušitel
- `Zadržení` - systém by měl být schopen zadržet narušitele (záleží na druhu objektu)

Správný zabezpečovací systém by měl být schopný odhalit narušitele, když se snaží dosáhnout svého cíle. To znamená, že kdykoliv se někdo pokusí vniknout do zabezpečené oblasti, tak se mu to nepodaří, aniž by ho systém detekoval.

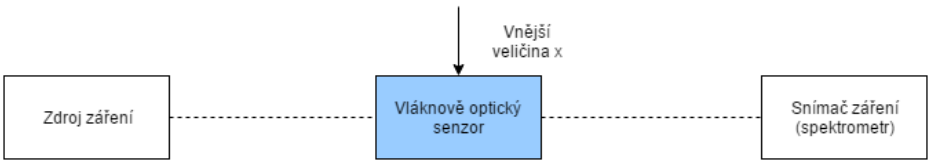
Klíčem k zabezpečení venkovního systému je použití vícezónového průchodu, kudy musí narušitel projít. Čím více je zón, kudy musí narušitel projít k dosažení svého cíle, tím je větší pravděpodobnost, že bude systémem detekován.

Na trhu neexistuje individuální nebo samostatná technologie zabezpečení, která by se sama postarala o zabezpečení prostoru. CCTV kamery, oplocení, jednotlivé senzory umístěné v zabezpečovaném prostoru, každý z těchto zabezpečovacích mechanismů hraje roli ve výsledném řešení. Důležité je určit, jakou úroveň zabezpečení bude náš systém poskytovat, což bývá v praxi největší problém.

Proto návrh zabezpečovacího systému vyžaduje pečlivou analýzu a vhodný návrh zabezpečovacích mechanizmů pro konkrétní prostor, který má být zabezpečen. To znamená správně specifikovat jednotlivé zabezpečovací prvky a vědět, jakou roli budou mít v zabezpečovacím systému. Samozřejmě každá oblast vyžaduje specifický a unikátní návrh zabezpečení. Je nutné mít na paměti, že navrhneme-li systém s co největším počtem zón, tím bude spolehlivěji detekovat narušitele, ale budou vyšší pořizovací náklady a náklady na provoz zabezpečovacího systému.

2.2 Optické vláknové senzorové sítě

Optické vláknové senzorové sítě se skládají ze zdroje záření, optických vláken a přijímače záření. Na obrázku č. 2.1 je blokové schéma optické vláknové senzorové sítě. Vnější veličina x deformuje signál ze zdroje záření, který je potom vyhodnocován ve snímači záření [14].



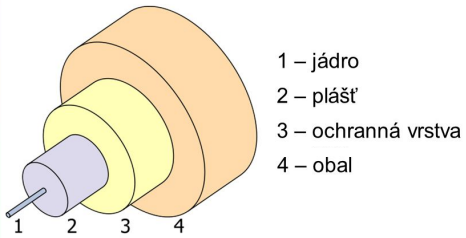
Obrázek 2.1: Blokové schéma optické sítě

2.2.1 Optické vlákno

Optická vlákna se v dnešní době nejčastěji používají v telekomunikacích pro rychlý přenos datového signálu na velké vzdálenosti, v tzv. metropolitních sítích (MAN). Optická vlákna se však používají i jako senzory v průmyslové oblasti, obzvláště v průmyslu zabývajícím se zabezpečením objektů.

2.2.1.1 Popis optického vlákna

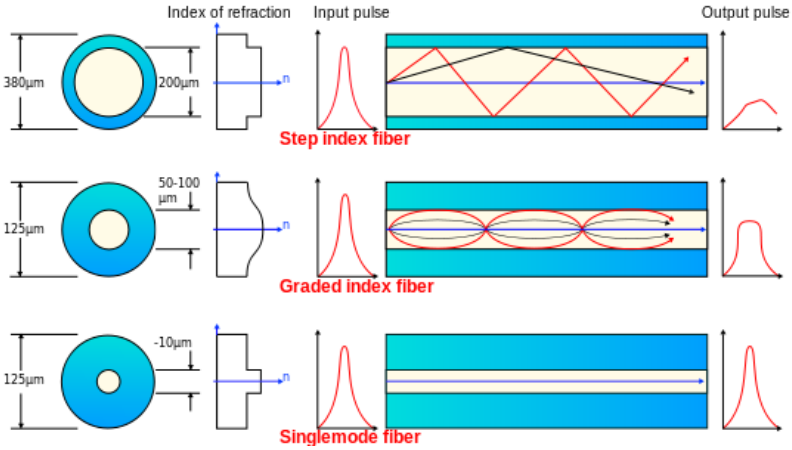
Optické vlákno je válečkový dialektrický vlnovod, ve kterém se šíří elektromagnetické vlny (zpravidla světlo či infračervené záření) ve směru osy vlákna s využitím principu totálního odrazu na rozhraní dvou prostředí s rozdílným indexem lomu. Vnitřní část se nazývá jádro, okolo jádra je plášť a primární ochrana. Obrázek č. 2.2 popisuje strukturu optického vlákna.



Obrázek 2.2: Průřez optickým vláknem [2]

2.2.1.2 Základní parametry optického vlákna

- `Průměr jádra (jednovidové)` $9\mu m$
- `Průměr jádra (vícevidové)` $62.5\mu m$
- `Průměr pláště` $125\mu m$
- `Průměr primární ochrany` $250\mu m$



Obrázek 2.3: Typy optických vláken [2]

`Vícevidové vlákno`

Vícevidové optické vlákno (multimode) je druh optického vlákna, které je nejčastěji používáno pro komunikaci na krátké vzdálenosti. Toto vlákno má větší průměr jádra než jednovidové optické vlákno. Dělí se podle šíření paprsků vláknem na vícovidové vlákno se skokovým indexem lomu (do vlákna vstupují vidy pod mnoha úhly) a gradientní vlákno (index lomu se zmenšuje se vzdáleností od středu vlákna). Názorně jsou druhy optických vláken ukázany na obrázku č. 2.3 [2].

`Jednovidové vlákno`

Jednovidové optické vlákno (single mode) je druh optického vlákna, který je používán pro přenos dat na větší vzdálenosti.

`Index lomu`

Index lomu vyjadřuje změnu rychlosti šíření světla při přechodu mezi různými prostředí.

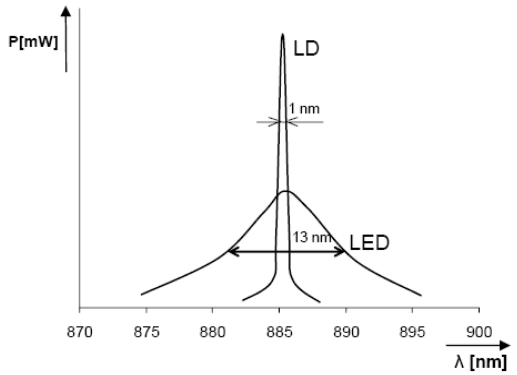
`Dokonalý odraz`
Když se světlo pohybuje v hustém (těžko proniknutelném) prostředí a dopadá na rozhraní pod širokým úhlem (větší než mezní úhel), světlo bude kompletně odraženo.

2.2.1.3 Optické vlákno jako senzor

Optická vlákna jsou v dnešní době převážně používána na přenos dat, kde některé jevy jako deformace vlákna v tahu, tlaku mají nepříznivý vliv na kvalitu přenášeného signálu. Avšak díky těmto jevům můžeme optické vlákno použít jako senzor. Existují dva druhy optických vláken, jedna skupina je určena pro přenos dat, druhá skupina optických vláken je určena pro použití jako optické senzory. Optické senzory jsou optická vlákna, která mají již od výroby záměrnou citlivost na snímanou veličinu. Například tah, tlak a podobně. Tato vlákna tvoří optické přenosové prostředí.

2.2.2 Zdroj záření

Zdroje optického signálu lze charakterizovat jako prvky, které generují optické záření. Nejčastěji se používají elektroluminiscenční diody (LED) nebo polovodičové laserové diody (LD). Světelná energie (tok fotonů) se generuje u diod LED mechanismem spontánní emise - fotony se generují v oblasti P-N nezávisle. U LD diody jsou čelní plochy krystalu zabroušené a tvoří nepropustné a polopropustné zrcadlo - tzv. Fabry-Petronův rezonátor. Rozměry rezonátoru jsou rovny celým násobkům poloviny vlnových délek záření. Za této podmínky dojde k optické rezonanci, zesílení světla a z malé čelní plochy vystupuje intenzivní monochromatické, koherentní záření. LD vykazuje na svém výstupu podstatně větší optické výkony než LED. Spektrální čistota (koherence) u LD je o několik řádů vyšší než LED, které zabírají širší pásmo kolem jmenovité vlnové délky λ. Na grafu č. 2.4 je znázorněna spektrální čistota LED a LD.



Obrázek 2.4: Porovnání spektrální čistoty LED a LD záření [14]

2.2.3 Snímač záření

Snímač je funkční prvek tvořící vstupní blok měřícího řetězce. Podstatou optoelektronických snímačů je snímat fyzikální veličinou vyvolaný optický jev, který vzniká mezi zdrojem a snímačem záření v optickém prostředí. Vliv snímané fyzikální veličiny způsobuje změnu (modulaci) přenášeného optického signálu. Všeobecně se využívají známé fyzikální jevy, které se vyskytují při interakci mezi různými fyzikálními poli a optickým prostředím. Při této interakci nastává změna vlatností optického prostředí, které se projeví jako změna optického signálu.

Pro měření změny optického signálu se používají optické vláknové snímače, které se dělí na: amplitudové, fázové, polarizační, s modulací vlnové délky, s modulací časového rozšíření impulzu. Pro účely snímání optického signálu se nejčastěji používají snímače s amplitudovou modulací.

`Optické vláknové snímače s amplitudovou modulací`
Modulace intenzity záření se realizuje změnou indexu lomu nebo změnou koeficientu útlumu. Útlum se mění například nepatrnou deformací optického vlákna, při které dochází k rozptylu vidů vyšších řádů do pláště vlákna.

2.3 Snímání signálu

V předchozí části je popsána struktura optické senzorové sítě a její princip činnosti. V této části bude uvedena konkrétní technika snímání vnější fyzikální veličiny a její detekce v takovéto síti.

2.3.1 Mřížkové vláknové optické senzory

FBG (fiber bragg grating) mřížky se skládají z periodických změn indexu lomu uvnitř jádra optického vlákna. Tato struktura se vyznačuje velmi úzkou selektivní odrazivostí se špičkou odražené vlnové délky λ_b, která je dána vztahem:

$$\lambda_b = 2n_{eff}\wedge \tag{2.1}$$

kde n_{eff} je efektivní index lomu vedeného módu ve vlákně a $\wedge$ je perioda indexu lomu modulovaná vztahem:

$$n(z) = n_{co} + \delta n[1 + cos(2\pi z/\wedge)] \tag{2.2}$$

kde n_{co} je index lomu neexponovaného jádra a δn je velikost foto-indukovaného indexu lomu. Tato struktura umožňuje dopředně vedené vidy (módy) odrážet a spojovat s odraženými vidy (módy).

Tyto mřížky se vyrábějí z fotosenzitivních germanio-křemičitých vláken. Mřížky dosahují délku 5mm s odrazivostí až 100% a šířkou odrazného spektra < 0.5 nm.

Princip FBG snímačů je zakódování měřené veličiny do odražené vlnové délky FBG mřížky. Hlavní prací FBG snímače je vyšetřování odražených vlnových délek. Toto vyšetření

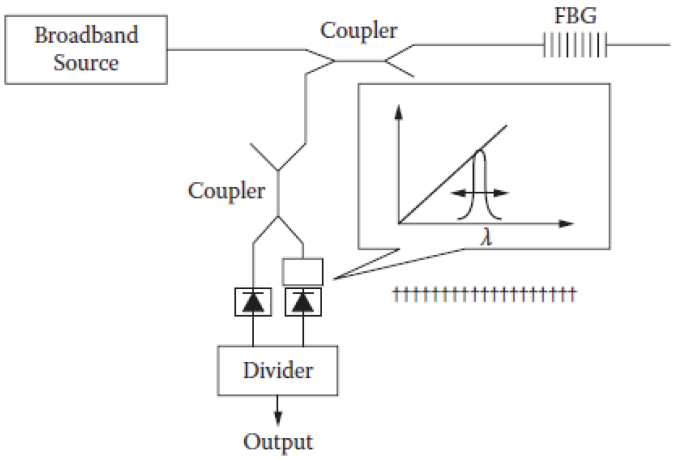
se dá provést jednoduchou metodou za pomoci spektrometru, avšak pro praktické aplikace, vzhledem k velikosti, ceně a nedostatku odolnosti spektrometru, je tato metoda nevhodná. Tudíž je nutné převést vlnovou délku na snadno měřitelný parametr jako je amplituda, frekvence nebo fáze. V dalších částech si představíme řešení převodů na amplitudu a frekvenci, které se používají v praxi.

2.3.2 Konverze vlnové délky na amplitudu

Měření amplitudy je v senzorických sítích nejčastěji používaná metoda. Převod vlnové délky na změnu amplitudy se dá zařídit velice jednoduše a v porovnání se spektrometry je výrazně cenově výhodnější.

2.3.2.1 Hranový filtr

Také nazývaný edge filtr je propustný filtr, který zachovává lineární vztah mezi změnou vlnové délky a výstupní intenzitou filtru. Typické blokové schéma filtru je znázorněno na obrázku č. 2.5. Odražené světlo FBG mřížky je rozdělelno do dvou paprsků se stejnou intenzitou. Jeden paprsek je filtrován před detekcí a druhý paprsek slouží jako referenční (není filtrován před detekcí). Tyto paprsky jsou potom zesíleny a navedeny do analogového děliče. Poměr filtrovaného a referenčního paprsku poskytuje informaci o vlnové délce, aniž by došlo k odchylkám intenzit vlivem výkyvů výkonu zdroje a ztrát ve spojeních optických vláken.

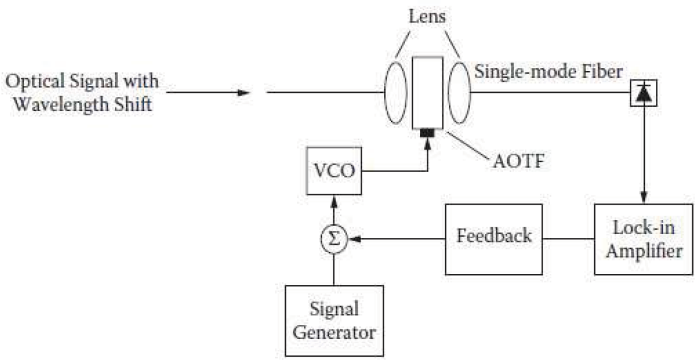


Obrázek 2.5: FBG senzor s hranovým filterm [14]

2.3.3 Konverze vlnové délky na frekvenci

Akusticko-optické laditelné filtry (AOTF) ukazují závislost propouštěných vlnových délek na řídící rádiové frekvenci, která byla použita. Posun vlnové délky může být sledován pomocí odchylky řídící rádiové frekvence. Blokové schéma je zobrazeno na obrázku č. 2.6. Pokud filtr na střední vlnové délce nesouhlasí s měřenou vlnovou délkou, systém zpětné vazby ohlásí chybu a aplikuje ji na napětím řízené oscilátory (VCO) pro nastavení frekvence. Tato

technika může být použita k detekci posunu vlnové délky z odráženého nebo přeneseného světla.



Obrázek 2.6: FBG senzor využívající AOTF

2.4 Předzpracování signálu

Zde je popsán preprocessing naměřeného signálu a prováděné filtrace, ze kterých získáme vstupní data pro diplomovou práci. Jinak řečeno, tato diplomová práce se zabývá zpracováním již předzpracovaných dat.

2.4.1 Raw data

Ze senzorické optické sítě máme k dispozici převedenou vlnovou délku světla na amplitudu. Tato data jsou digitalizovaným výstupem hranového filtru. V rámci diplomové práce jsou tato data brána jako vstupní signál pro další filtry, které jsou popsány níže.

2.4.2 Horní a dolní propust (high and low pass)

K filtraci je použitý diskrétní digitální filtr IIR (infinite impulse response), který má nekonečnou impulsní odezvu a vyžaduje vždy minimálně jednu zpětnouvazební smyčku. IIR je rekursivní filtr, kde přenos je tvořen podílem polynomů. Výhodou použití tohoto filtru je rychlá odezva na vstupní data, ovšem nevýhodou jsou problémy se stabilitou.

Vhodným nastavením tohoto filtru získáme horní či dolní propust. Poté aplikujeme tyto filtry na výstupní signál optické senzorické sítě a získáme oddělené signály s nízkými frekvencemi a samostatný signál s vysokými frekvencemi.

2.4.3 FFT (Fast Fourier transform)

Fourierova transformace je integrální tranformace převádějící signál mezi časově a frekvenčně závislým vyjádřením pomocí harmonických signálů. Slouží pro převod signálů z časové oblasti do oblasti frekvenční. Pro výpočet je použitá diskrétní fourierova transformace.

Diskrétní fourierova transformace se využívá v případě číslicového zpracování signálu. Poté pracujeme s konečným počtem hodnot a to jak v časové, tak ve frekvenční oblasti. Signál má v obou oblastech stejný počet hodnot N. V literatuře je tato transformace označována jako DFT.

Uvažujeme tedy komplexní řadu $x[n]$ o N prvcích

$$x_0, x_1, x_2, x_3, \ldots, x_{N-1} \tag{2.3}$$

kde každé x je komplexní číslo

$$x_i = x_{Re} + j \cdot x_{Im} \tag{2.4}$$

Za předpokladu, že řada vně rozsahu 0 až $N-1$ je periodické prodloužení intervalu, pak můžeme definovat DFT jako

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-jn\theta_k} = \sum_{n=0}^{N-1} x[n]e^{\frac{-j2\pi nk}{N}}, k = 0, 1, \ldots, N-1 \tag{2.5}$$

pak zpětná transformace je

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{jn\theta_k} = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{\frac{j2\pi k}{N}}, n = 0, 1, \ldots, N-1 \tag{2.6}$$

FFT využívá symetrie a redundance vstupního signálu. Hlavní výhodou FTT je značná rychlost oproti DFT. Pro výpočet N hodnot pomocí DFT je potřeba provést N^2 komplexních násobení a $N(N-1)$ komplexních sčítání, takže asymptotická složitost je $O(N^2)$. Naproti tomu asymptotická složitost výpočtu FTT je pouze $O(\frac{N}{2} \cdot log_2(N))$.

2.4.4 WVD (Wigner Ville distribution)

Mějme daný signál $x[t]$ jehož autokorelační funkce je dána:

$$C_x(t_1, t_2) = \langle (x[t_1] - \mu[t_1])(x[t_2] - \mu[t_2])^* \rangle \tag{2.7}$$

kde $\langle \ldots \rangle$ značí průměr všech možných realizací, μ je průměr, která může být či nemusí být funkcí času. Wigner funkce $W_x(t, f)$ je dána prvním výrazem autokorelační funkce v termech průměrného času $t = \frac{(t_1+t_2)}{2}$ a časového zpoždění $\tau = t_1 - t_2$ a zpožděním Fourierovy transformace.

$$W_x(t, f) = \int_{-\inf}^{\inf} C_x(t + \frac{\tau}{2}, t - \frac{\tau}{2})e^{-2\pi i\tau f} d\tau \tag{2.8}$$

Pro jednotlivý signál je Wignerova funkce dána následovně:

$$W_x(t, f) = \int_{-\inf}^{\inf} x(t + \frac{\tau}{2})x(t - \frac{\tau}{2})^* e^{-2\pi i\tau f} d\tau \tag{2.9}$$

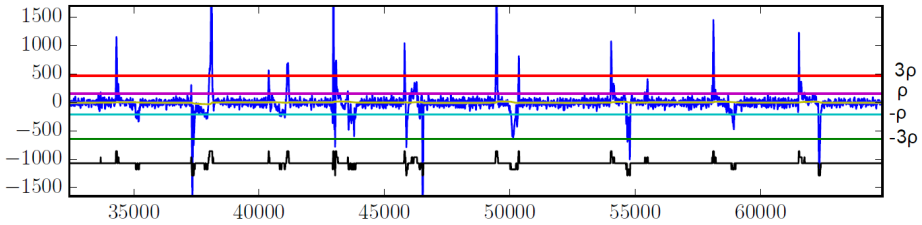
Důvod k používání Wignerovy funkce je takový, že redukuje šum a přesto je plně ekvivaletní s nestacionární autokorelační funkcí.

2.5 Detekce anomálií

Zpracování vlnových délek za pomocí filtrů nám poskytuje signál, který popisuje vliv některé z měřených veličin na optickou senzorovou sít. Nyní je však potřeba zajistit, abychom byli schopni správně detekovat, že vznikla anomálie (událost vyvolaná vnější fzikální veličinou) a nejedná se třeba o vzniklý šum v síti.

Jako detektor anomálie v síti se používá statistická prahová metoda. Jedná se o nejjednodušší metodu pro detekování anomálie, kde se nastaví hodnota prahu. Jestliže měřený signál překročí tuto hranici, pak je detekována anomálie. Hodnota prahu musí být nastavena přesně, pro správnou funkčnost detektoru. Pokud je nastavena vysoko, tak detektor nedetekuje anomálii, i když tato anomálie v síti vznikla. Na druhou stranu, když je hodnota nastavena moc nízko, tak potom mohou vznikat takzvané falešné poplachy, které jsou způsobeny pouze šumem signálu. Je tedy nutné nastavit hranici těsně nad úroven špiček šumu.

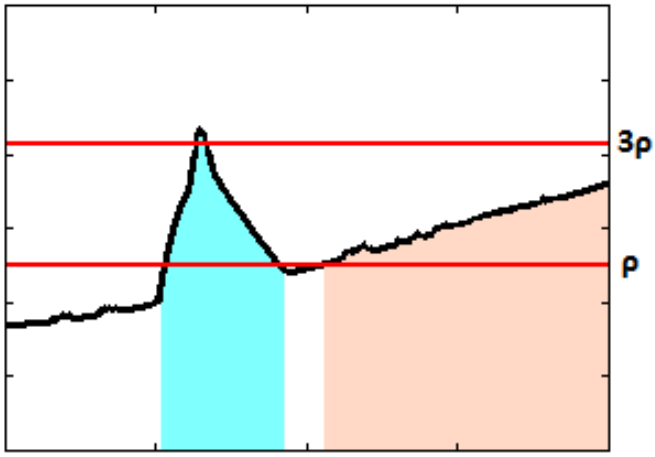
Anomálie v experimentálních datech použité pro tuto práci jsou detekovány pomocí metody dvou prahů. Jeden s hodnotou ρ a druhý s hodnotou 3ρ. Na obrázku č. 2.7 jsou znázorněny použité prahy. Mezi prahy ρ a $-\rho$ se nachází šum. V tomto pásmu víme, že nebude detekována žádná anomálie. Pokud signál překročí prahy 3ρ a -3ρ, pak je detekována anomálie, která je dána rozsahem, kde signál překročil prahy ρ a $-\rho$. Tento princip se nazývá hysterézní prahování. Ukázka detekce anomálie je na obrázku č. 2.8.



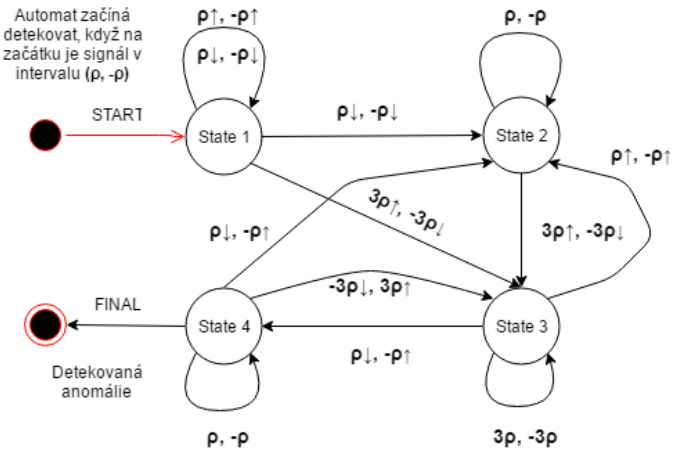
Obrázek 2.7: Ukázka detekce anomálií v signálu

Detekce anomálie probíhá podle následujícího nedeterministického automatu popsaném na obrázku č. 2.9. Automat začne vyhodnocovat vstupní signál, který se na začátku pohybuje v šumovém pásmu (v intervalu $(\rho, -\rho)$). Pokud signál přejde do některého z intervalů mezi prahy $(\rho, 3\rho)$ nebo $(-\rho, -3\rho)$ tak přejde do dalšího stavu, ve kterém se dále vyhodnocuje průběh signálu. Pokud signál překročí práh (3ρ) nebo (-3ρ), pak se změní stav a v tento okamžik již víme, že se jedná o anomálii v signálu, která končí, když signál přejde zase do šumového pásma. V opačném případě se automat vrátí zase do počátečního stavu.

Tato metoda detekce anomálií je aplikována na předzpracovaný signál. Tudíž v každém nově získaném kanálu máme specifické hodnoty prahů, pro které následně detekujeme anomálie.



Obrázek 2.8: Metoda hysterézního prahování - ve výsledku je detekovaná pouze modrá oblast signálu



Obrázek 2.9: Graf přechodu automatu pro detekci anomálií v signálu

Kapitola 3

Data mining

Data mining neboli vytěžování informací je analytická metodologie získávání netriviálních skrytých a potenciálně užitečných informací z dat. Data mining nachází velké uplatnění ve vědeckých výzkumech, marketingu nebo pro monitorování podezřelých aktivit a odhalení potencionálních škůdců na internetu [1].

Metodologie data miningu je vždy stejná a dá se popsat v následujících šesti krocích, které jsou znázorněny na obrázku č. 3.1. Popis jednotlivých kroků je následující:

- `Pochopení problému:` bez porozumění požadavků zákazníka a jasného stanovení cíle se neobejde žádný projekt. Stejně tak to platí i v data miningu. V této fázi dochází k návrhu a tvorbě plánu pro řešení daného problému.

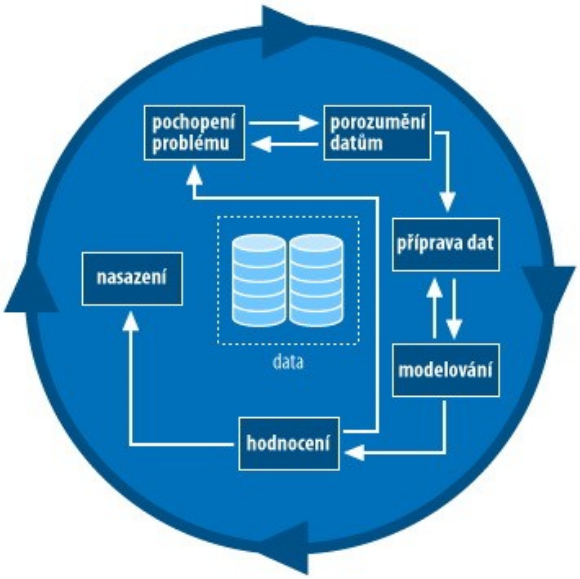
- `Porozumění datům:` je nezbytné pro další vývoj procesu. V této fázi také dochází k vytváření prvních hypotéz, které se v průběhu celého procesu snažíme potvrdit. Někdy však můžeme hypotézu vyvrátit nebo naopak potvrdit.

- `Příprava dat:` zde dochází k integraci více datových zdrojů, čištění a úpravě dat do podoby, kterou vyžadují jednotlivé analyzující algoritmy. Tento proces nelze správně provést bez znalosti dat. Špatná integrace by mohla vyústit ke znehodnocení zdrojů dat a ovlivnit celkové kvality řešení.

- `Modelování:` obsahuje testování vhodných metod a nastavení jejich parametrů pro řešení definovaného problému. Z tohoto kroku vybíráme několik nejlepších získaných řešení, které postupují do dalšího kroku.

- `Hodnocení:` v této fázi dochází ke konečnému hodnocení a selekci získaných modelů podle různých vlastností a ověření správnosti získaných řešení za pomoci těchto modelů. Dle získaných výsledků je již možno zvážit případnou implementaci celého procesu.

- `Nasazení:` je posledním krokem v celém procesu. Je však nutné podotknout, že proces nekončí, ale začíná se cyklicky opakovat.



Obrázek 3.1: Schéma metodologie data miningu

3.1 Klasifikace

Klasifikace je data miningová technika založená na strojovém učení, která řeší druh problému, kdy máme určit, do které z kategorií dat dané pozorování patří. K tomu nám slouží testovací množina dat obsahující pozorování (data, instance), pro která jsou kategorie určeny správně. Jednotlivá pozorování jsou převedena do množiny kvantifikovatelných vlastností (příznakových vektorů). Tyto hodnoty mohou být kategorické, reálné nebo celočíselné hodnoty. V terminologii strojového učení je klasifikace považována za metodu učení s učitelem, to zamená učení, při kterém je známá testovací množina správně klasifikovaných příkladů. Celkové schéma přístupu této metody je, že se nejdříve naučí (natrénuje) klasifikátor na základě testovacích dat. Hotový klasifikátor se pak používá pro klasifikaci nových dat.

3.2 Shlukování

Shluková analýza je vícerozměrná statistická metoda, která se používá ke klasifikaci objektů. Slouží ke třídění jednotlivých objektů do skupin (shluků) tak, aby si jednotlivé objekty náležící do stejné skupiny byly podobnější než objekty z ostatních skupin. Hierarchické shlukování vytváří systém podmnožin, kde průnikem dvou podmnožin - shluků je buď prázdná množina, nebo jedna z nich. Pokud nastane alespoň jednou druhý případ, je systém hierarchický. Tedy jedná se o zjemňování klasifikace. Mezi nejznámější přístupy, jak shlukovat objekty na základě jejich vzdálenosti nebo podobnosti patří například metody nejbližšího souseda, nejvzdálenějšího souseda, centroidní metoda atd.

3.3 Pochopení problému

Tato diplomová práce se zabývá detekcí anomálií v optické senzorové síti a zároveň je kladen požadavek na klasifikaci jednotlivých anomálií, které v síti byly detekovány. Jelikož se jedná o data, která na první pohled nevykazují podobné vlastnosti, je potřeba vhodnými metodami předpřipravit data pro další zpracování a následně je přiřadit do konkrétní třídy.

3.4 Porozumění datům

Pro účely této diplomové práce byla data z optické senzorové sítě poskytnutná od společnosti Safibra s.r.o., která vyvíjí a vylepšuje tyto senzorické sítě. K dispozici této práce byl poskytnut záznam experimentu s jedním optickým vláknem, které bylo zakopáno v zemi. Následně pak se zkoumal vliv kroků osoby po povrchu nebo pád těžkých předmětů. Vzhledem k tomu, že tato síť je prototypem, který se stále vyvíjí, tak poskytnutná data jsou zašuměná a nestabilní (stejná anomálie vykazuje jiný charakter v naměřených datech). Dále byl poskytnut videozáznam, který je časově synchronizován se záznamem z optické senzorové sítě. Tento video záznam umožňuje pochopit, co se v daný okamžik ve skutečnosti událo.

3.5 Příprava dat

Příprava dat, jejich filtrace a následná detekce anomálií je popsána v kapitolách 2.4 a 2.5.

Podle poskytnutného video záznamu bylo možné manuálně oklasifikovat detekované anomálie. Takže tím získáváme pro další experimentování množinu oklasifikovaných dat, které je poté možné použít jako trénovací data pro algoritmy využívající trénovací množinu dat. Následně pak můžeme zároveň testovat, zda daný algoritmus klasifikuje nebo shlukuje data stejných/podobných anomálií správně.

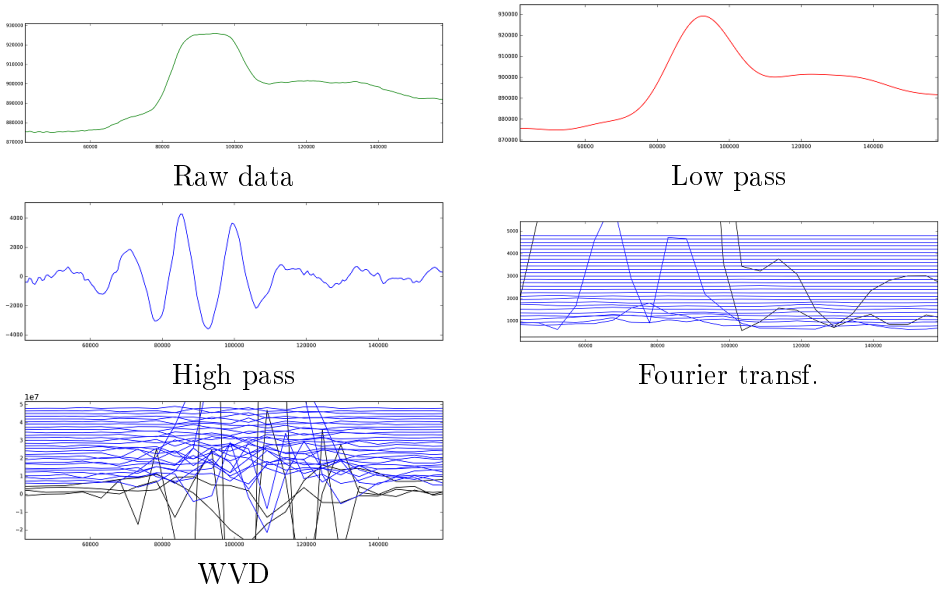
3.6 Modelování

Nejprve je nutné analyzovat data, pro která budeme hledat vhodné metody pro datamining. K dispozici máme časové série z aplikovaných filtrů, které obsahují jeden průběh. Oproti tomu výstupy z Fourierovy transformace a Wigner Ville distribuce jsou posloupnosti zastoupení jednotlivých frekvenčních složek ve vstupním signálu.

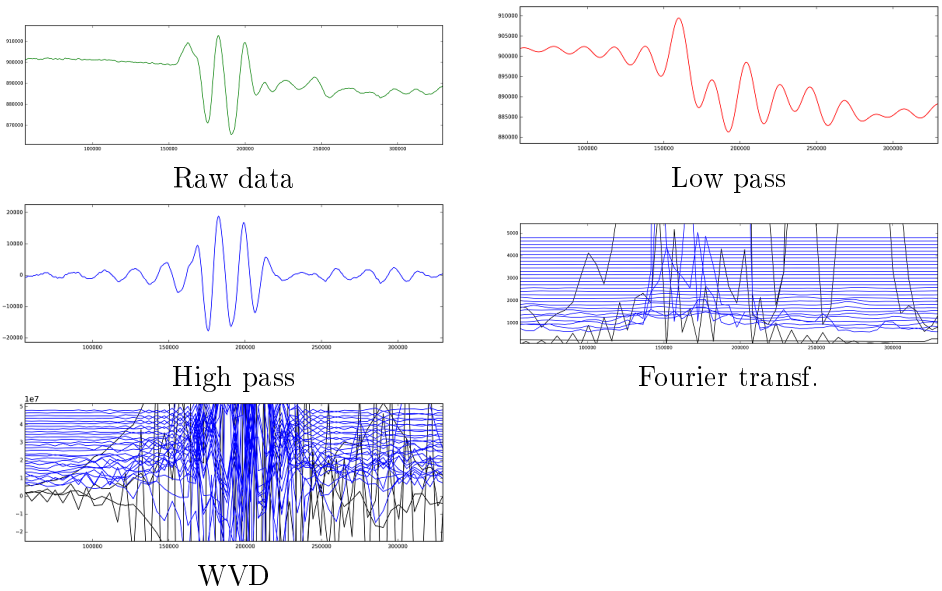
Jako příklad si uvedeme grafickou reprezentaci vstupních dat dvou podobných anomálií, mezi kterými musíme najít charakteristické společné vlastnosti, které bude zapotřebí rozlišovat při dataminingu. Na vzorku č. 3.2 je detekována anomálie v průběhu WVD, konkrétně dva pády kovadliny. Na druhém vzorku č. 3.3 je zopakovaná tatéž anomálie. Jak je na první pohled vidět, najít mezi těmito anomáliemi podobnost nebude lehký úkol.

3.7 Množina rozpoznávacích algoritmů

Tato část bude zkoumat množinu rozpoznávacích algoritmů, nad kterými budeme provádět experimenty. Konkrétně se bude zabývat teoretickým rozborem funkce jednotlivých al-



Obrázek 3.2: 1. vzorek WVD detekované anomálie - pád kovadliny



Obrázek 3.3: 2. vzorek WVD detekované anomálie - pád kovadliny

goritmů. Pro jednotlivé metody jsou uvedeny pouze základní informace. Jejich konkrétní použití je popsáno v kapitole 5.

3.7.1 Prokládání křivek

Polynomická či polynomiální regrese představuje proložení (aproximaci) zadaných hodnot polynomem a jde o zvláštní případ lineární regrese. Koeficienty hledaného polynomu jsou

metodou nejmenších čtverců vypočteny tak, aby součet druhých mocnin odchylek původních hodnot od získaného polynomu byl minimální.

Cílem je proložit hodnotami $x_i, y_i, i = 1, \ldots, n$ polynom k-tého stupně $P_k(x) = p_0 + p_1 x + \ldots + p_k x^k$. Koeficienty $p_0, \ldots, p_k$ jsou přitom voleny tak, aby součet druhých mocnin odchylek

$$e_i \equiv y_i - P_k(x_i) \tag{3.1}$$

byl minimální tj.

$$F = \sum_{i=1}^{n} e_i^2 \to min. \tag{3.2}$$

Úloha vede na problém nejmenších čtverců.

3.7.1.1 Problém nejmenších čtverců

Dosazením hodnot x_i, y_i do polynomiálního modelu $y = P_k(x)$ přímo dostaneme aproximační problém. Z definice odchylky e_i zřejmě platí $y = P_k(x) + e_i$. Zde nám e_i reprezentuje chybu vzniklou při měření veličiny y_i, přičemž předpokládáme, že veličiny x_i jsou známy přesně.

V maticovém zápisu

$$Ax = b + e \tag{3.3}$$

kde

$$A = \begin{bmatrix} x_1^k & \ldots & x_1 & 1 \\ x_2^k & \ldots & x_2 & 1 \\ \vdots & & \vdots & \vdots \\ x_n^k & \ldots & x_n & 1 \end{bmatrix}, x = \begin{bmatrix} p_k \\ \vdots \\ p_1 \\ p_0 \end{bmatrix}, b = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, e = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix} \tag{3.4}$$

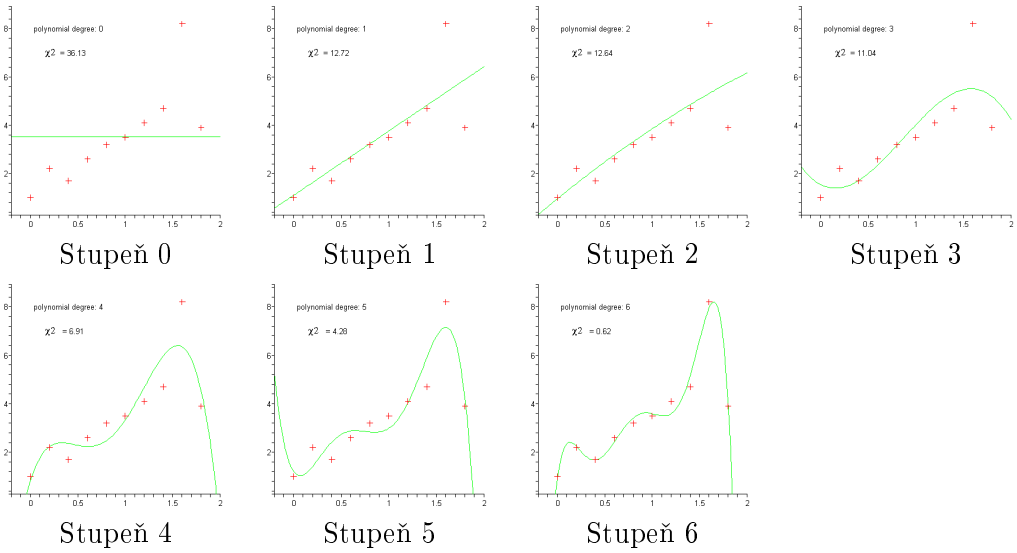
$p_0, p_1, \ldots, p_k$ jsou neznámé koeficienty hledaného polynomu a cílem je dosáhnout takového řešení, aby norma vektoru e byla minimální. Úloha se řeší metodou nejmenších čtverců.

Příklady proložení množiny bodů křivkou jsou uvedeny na obrázku č. 3.4. Jsou zde ukázány tvary křivek v závislosti na stupni polynomu.

3.7.2 Dynamic time wrapping (DWT)

Dynamic time wrapping (česky: borcení časové osy) je algoritmus pro měření podobnosti mezi dvěma sekvencemi signálu, které mohou mít rozdílné rychlosti. Například pomocí DWT můžeme detekovat pattern chůze, dokonce i když jedna osoba šla rychleji než ta druhá, dokonce i když během chůze různě zrychlovaly nebo zpomalovaly.

Cílem DTW metody je porovnat dvě (časově závislé) sekvence $X = (x_1, x_2, \ldots, x_N)$ délky $N \in \mathbb{N}$ a $Y = (y_1, y_2, \ldots, y_M)$ délky $M \in \mathbb{N}$. Tyto sekvence jsou diskrétní signály v čase. Dále musíme sestrojit *feature prostor* κ tak, že $x_n, y_m \in \kappa$ pro $n \in [1 : N]$ a $m \in [1 : M]$.

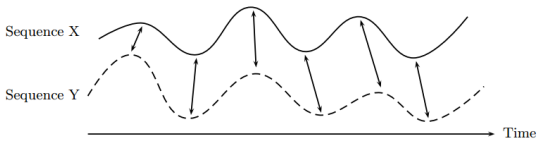


Obrázek 3.4: Ukázka polynomiální regrese pro polynom 0 až 6 stupně

Pro porovnání těchto dvou rozdílných sekvencí musíme spočítat *lokální vzdálenost*, které je definováno jako zobrazení

$$c : \kappa \times \kappa \rightarrow \mathbb{R}_{\geq 0} \tag{3.5}$$

Pokud $c(x, y)$ má malou hodnotu (nízká cena) když x a y mají podobnou hodnotu, naopak když $c(x, y)$ má velkou hodnotu (velká cena) když x a y jsou rozdílná. Tento výpočet lokální vzdálenosti provedeme provedeme pro každou hodnotu ze sekvencí X a Y. Dostaneme *matici vzdáleností* $C \in \mathbb{R}^{N \times M}$, která je definována jako $C(n, m) = c(x_n, y_m)$. Na obrázku č. 3.5 jsou sekvence, pro které spočteme matici vzdáleností, která je znázorněna na obrázku č. 3.6.

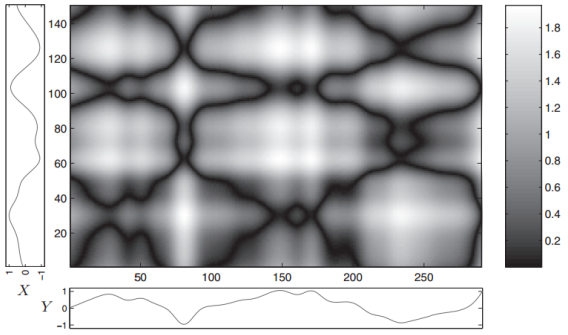


Obrázek 3.5: Vstupní sekvence pro metodu DWT

Nyní vytvoříme matici reprezentující *akumulované energie* D podobnou matici vzdáleností. Tato matice bude obsahovat minimální vzdálenosti ze specifického bodu do ostatních bodů. Matice akumulované energie je definována následovně

$$D(n, 1) = \sum_{k=1}^{n} c(x_k, y1) \quad \text{for} \quad n \in [1 : N] \tag{3.6}$$

$$D(1, m) = \sum_{k=1}^{m} c(x_1, y_k) \quad \text{for} \quad m \in [1 : M] \tag{3.7}$$



Obrázek 3.6: Matice vzdáleností dvou sekvencí

$$D(n,m) = min\{D(n-1,m-1), D(n-1,m), D(n,m-1)\} + c(x_n, y_m) \tag{3.8}$$

kde $1 < n \leq N$ a $1 < m \leq M$. Tato operace má časovou složitost $O(NM)$.

V matici akumulované energie D musíme najít *wraping path*, která má následující omezení:

- Cesta musí začínat v bodě $D(0,0)$ a končit v bodě $D(M,N)$.
- Cesta nemůže jít zpátky v čase, takže vede vždy dopředu. Tzn. že z bodu $D(i,j)$ můžeme jít pouze doprava $D(i+1,j)$ nahoru $D(i,j+1)$ nebo diagonálně $D(i+1,j+1)$.

Tyto podmínky zjednoduší problém na Dynamic Programming problém, který je řešitelný v čase $O(NM)$. [13]

3.7.3 Fázový prostor (phase space)

Předpokládejme, že daný signál je generován nějakým dynamickým systémem. Pak můžeme určitý stav tohoto systému popsat bodem ve fázovém prostoru (phase space). Tyto body zobrazené na časové ose vytvářejí trajektorii fázového prostoru. Z určitého pohledu můžeme tvrdit, že výsledná trajektorie je zobrazení našeho signálu dynamického systému je projekce do bodů se souřadnicemi ve fázovém prostoru. Podle [12] je možné z jakéhokoliv signálu vytvořit projekci do fázového prostoru. Nejpoužívanější metodou pro vytvoření projekce signálu do fázového prostoru je metoda časového zpoždění (method of time delay). Hlavním úkolem při používání této metody je určit hodnoty časového zpoždění τ a dimenzi m.

3.7.3.1 Metoda časového zpoždění (time delay method)

Metoda časového zpoždění je nejčastěji používaná metoda pro tvoření fázového prostoru. Pokud máme signál tvořený skalárními proměnnými, můžeme zkonstruovat vektor

$$x(t_i), \quad i = 1, \ldots, N \tag{3.9}$$

ve fázovém prostoru v čase t_i následovně:

$$X(t_i) = [x(t_i), x(t_i + \tau), x(t_i + 2\tau), \ldots, x(t_i + (m-1)\tau)] \tag{3.10}$$

kde $i = 1, \ldots, N - (m-1)\tau$, τ je časové zpoždění, m je dimenze tvořeného fázového prostoru a $M = N - (m-1)\tau$ je počet bodů ve fázovém prostoru.

3.7.3.2 Výběr časového zpoždění

K určení vhodné časového zpoždění používáme AMI (average mutual information), což je generalizovaná autokorelační funkce. AMI mezi jednotlivými naměřenými signály A a B je definován:

$$I_{AB} = \sum_{a_i b_j} P_{AB}(a_i, b_j) log_2[\frac{P_{AB}(a_i, b_j)}{P_A(a_i)P_B(b_j)}] \tag{3.11}$$

kde $P_A(a_i)$ je pravděpodobnost výskytu a_i v signálu A, $P_B(b_j)$ je pravděpodobnost výskytu b_j v signálu B a $P_{AB}(a_i, b_j)$ je přidružená pravděpodobnost výskytu a_i v signálu A a b_j v signálu B. Pokud $A = \{x(t_1), x(t_2), \ldots, x(t_i)\}$ a $B = \{x(t_1 + \tau), x(t_2 + \tau), \ldots, x(t_i + \tau)\}$, pak výsledný počet

$$I(\tau) = \sum_{x(t_i), x(t_i+\tau)} P(x(t_i), x(t_i + \tau)) log_2[\frac{P(x(t_i), x(t_i + \tau))}{P(x(t_i))P(x(t_i + \tau))}] \tag{3.12}$$

reprezentuje průměrnou obsaženou informaci, kterou víme o hodnotě x v čase $t+\tau$ z hodnoty x v čase t.

3.7.3.3 Výběr dimeze

Pro odhadování rozměru dimenze se používá metoda tzv. falešných nejbližších sousedů (FNN), tato metoda je popsána v [9]. Myšlenka metody je taková, že pokud trajektorie bodů zobrazená do malé dimenze kříží sama sebe, tak tomuto jevu říkáme výskyt falšných sousedů. Když budeme zvyšovat dimenzi fázového prostoru, tak počet překřížení trajektorie se bude zmenšovat, tím pádem i počet výskytů falešných sousedů. Když je dimenze dostatečně velká, tak vymizí výskyt falešných sousedů. K určení zda se jedná o falešného souseda či ne, potřebujeme vyhodnotit dvě kritéria zmíněná v [9]. Nejprve vypočteme hodnotu $R(t_i))$ definovanou následovně:

$$R(t_i)) = \frac{|x(t_i + m\tau) - x^{NN}(t_i + m\tau)|}{\|X_m(t_i) - X_m^{NN}(t_i)\|} \tag{3.13}$$

Jako prahovou hodnotu R_T zvolíme 10 a 15 [6]. Pokud $R(t_i) \geq R_T$ tak tento stav považujeme za falešného souseda. Jako další kritérium falešnosti sousedství použijeme následující zlomek

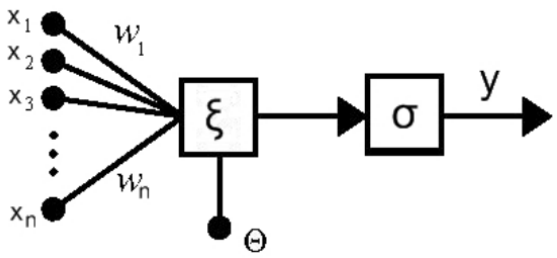
$$\frac{|x(t_i + m\tau) - x^{NN}(t_i + m\tau)|}{R_A} \geq A_T \tag{3.14}$$

kde R_A je radius definovaný

$$R_A^2 = \frac{1}{N} \sum_{i=1}^{N} [x(t_i) - \overline{x}]^2, \quad \overline{x} = \frac{1}{N} \sum_{i=1}^{N} x(t_i) \tag{3.15}$$

3.7.4 Neuronové sítě

Neuronové sítě jsou jedním z výpočetních modelů používaných v umělé inteligenci. Stejně jako mnoho jiných informatických odvětví, tak i umělé neuronové sítě získaly prvotní inspiraci v biologické předloze. Základním stavebním kamenem každé neuronové sítě je perceptron. Na obrázku č. 3.7 je znázorněn perceptron. Nejprve je dáno n vstupů $x_1, x_2, \ldots, x_n$, které jsou ohodnoceny n váhami $w_1, w_2, \ldots, w_n$. Dále θ značí vnitřní práh, ξ označuje vnitřní potenciál a σ aktivační funkci. Na konci je y, které označuje výstup perceptronu.



Obrázek 3.7: Perceptron

Perceptron začne pracovat tak, že z daných vstupů vytvoří výstupní potenciál

$$\xi = \sum_{i=1}^{n} w_i x_i \tag{3.16}$$

Poté aktivační funkce za pomocí θ vyhodnotí, zda-li je ξ dostatečně vysoký. Toto je nejjednodušší interpretace aktivační funkce, kde je perceptron buďto aktivní (na výstupu je 1) či nikoliv (na výstupu je 0). Formálně zapsáno:

$$\sigma(\xi) = \begin{cases} 1, & \xi \geq \theta \\ 0, & \xi < \theta \end{cases} \tag{3.17}$$

Pro výstup perceptronu y platí:

$$y = \sigma(\xi) \tag{3.18}$$

Neuronová síť se tedy skládá z perceptronů, které jsou vzájemně propojovány tak, že výstup neuronu může být zároveň vstupem dalších perceptronů. Pro popis modelů neuronových sítí používáme pojmy:

- `Konfigurace` je vektor vah všech perceptronů tvořící síť.
- `Topologie` označuje architekturu sítě - kolik má perceptronů, jak jsou pospojovány apod.
- `Aktivní fáze` popisuje činnost od konkrétního vstupu sítě až k dosažení výstupu.
- `Adaptivní fáze` značí změnu konfigurace sítě - proces učení.

3.7.4.1 Příklad vícevrstvé sítě

Klasická topologie se skládá ze 3 vrstev perceptronů. První vrstva je vstupní, druhá skrytá a třetí výstupní. Každý perceptron z nižší vrstvy je spojen se všemi neurony vrstvy následující. Nechť vstupní vrstva se skládá z 26 perceptronů, počet skrytých perceptronů je variabilní a výstupní vrstva obsahuje 10 neuronů. Jako aktivační funkce je použita standardní sigmoida daná předpisem:

$$\sigma(\xi) = \frac{1}{1 + e^{(-\xi)}} \tag{3.19}$$

Aktivní fáze provádí výpočet pro zadané vstupy $x_1, x_2, \ldots, x_{26}$ postupuje po jednotlivých vrstvách. V každé vrstvě probíhá výpočet postupně pro všechny perceptrony. Po zkompletování výstupů všech perceptronů jedné vrstvy se postupuje k vrstvě následující. Výstup poslední vrstvy $y_1, y_2, \ldots, y_{10}$ je výstupem celé sítě. Adaptivní fáze pro vzory ve tvaru $x_1, x_2, \ldots, x_{26}$, $y_1, y_2, \ldots, y_{10}$ se použije učící algoritmus tzv. backpropagation. Pro vstupy $x_1, x_2, \ldots, x_{26}$ každého vzoru, jsou vypočteny reálné výstupy $y_1, y_2, \ldots, y_{10}$. Poté dojde k výpočtu aktuální chyby sítě k množině vzorů:

$$E(w) = \sum_{k=1}^{p} E_k(w) \tag{3.20}$$

kde w značí aktuální konfiguraci vah sítě a p počet vzorů. Celková chyba sítě odpovídá chybám jednotlivých vzorů. Chyba k-tého vzoru odpovídá:

$$E_k(w) = \frac{1}{2} \sum_{j \in Y} (y_j - d_j)^2 \tag{3.21}$$

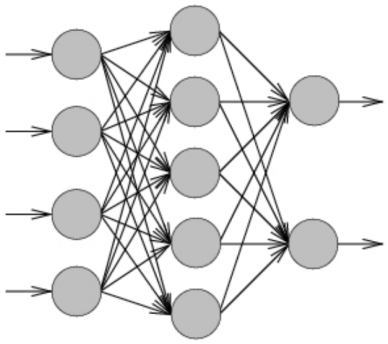
kde Y je množina všech perceptronů ve výstupní vrstvě, y značí reálné výstupy a d očekávané výstupy pro k-tý vzor. Minimalizace chyby a adaptace jednotlivých vah probíhá pro každou váhu následovně:

$$w^{(t)} = w^{(t-1)} - \delta w^{(t)} \tag{3.22}$$

kde $w^{(t)}$ označuje novou váhu na spoji mezi dvěma perceptrony a $w^{(t-1)}$ původní váhu. Přírůstek $\delta w^{(t)}$ je počítán jako diferenciál funkce celkové chyby sítě vzhledem k dané váze. Detailněji popsáno v publikaci [11].

3.7.5 Algoritmy podpůrných vektorů (SVM)

Support vector machine neboli algoritmy podpůrných vektorů tvoří určitou kategorii tzv. jádrových algoritmů (kernel machines). Tyto metody se snaží využít výhody poskytované efektivními algoritmy pro nalezení lineární hranice a zároveň jsou schopny reprezentovat vysoce složité nelineární funkce. Jedním ze základních principů je převod daného původního prostoru do jiného, vícedimensionálního, kde již lze od sebe oddělit třídy lineárně. [5] Na obrázku č. 3.9 je znázorněn původní dvourozměrný prostor, kde jsou dvě třídy, oddělené nelineárně kružnicí. Přidáním další dimenze vznikne možnost prvkům třídy uvnitř kružnice přidat další souřadnice, která je posune např. nahoru podél nové osy x_3, takže pro oddělení



Obrázek 3.8: Vícevrstvvá perceptronová síť

obou tříd již lze použít rovinu rovnoběžnou s rovninou danou osami x_1 a x_2. Optimální lineární oddělovač se v algoritmu SVM hledá pomocí metody kvadratického programování. Kvadratické programování je obdoba s hledáním maxima jako u lineárního programování, problém je však složitjší. Předpokládejme, že jsou příklady x_i s klasifikací $y_i = \pm 1$ a cílem je najít optimální oddělovač. Problém lze převést na hledání hodnot parametru α_i, které maximalizují výraz

$$\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i x_j) \tag{3.23}$$

přičemž platí omezení

$$\alpha_i \geq 0 \quad a \quad \sum_i \alpha_i y_i = 0 \tag{3.24}$$

Výraz má dvě významné vlastnosti. Má jediné globální maximum, které lze efektivně najít, a dále datové body do výrazu vstupují ve formě skalárního součinu jednotlivých dvojic. Tato druhá vlastnost platí i pro rovnici lineárního oddělovače. Jakmile jsou spočteny optimální parametry α_i, je oddělovač dán rovnicí

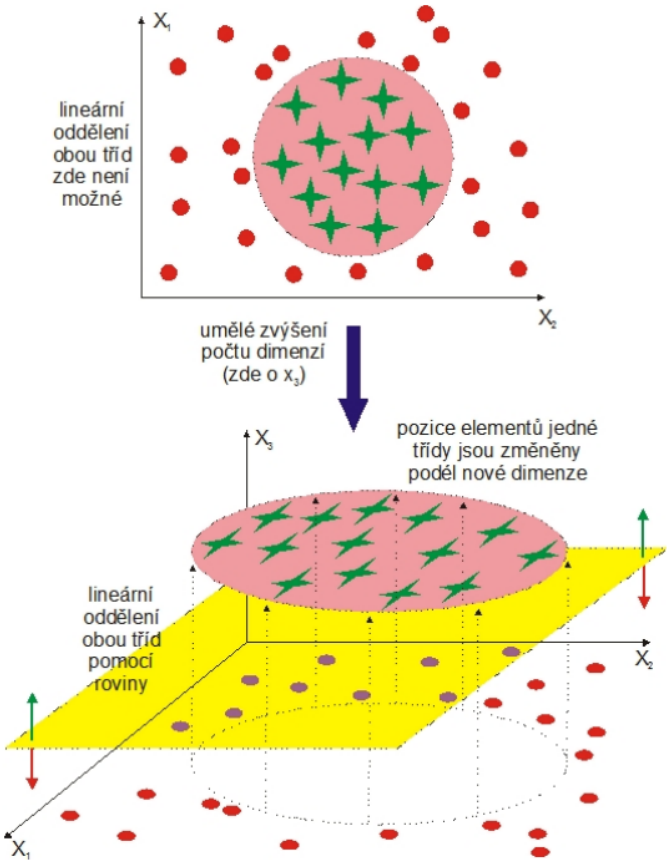
$$h(x) = sign(\sum_i \alpha_i y_i (x x_i)) \tag{3.25}$$

Vzhledem k omezením při hledání α_i platí, že lineární oddělovač má nenulové váhy α_i pro každý datový bod kromě těch bodů, které jsou nejblíže vlastnímu oddělovači. Tyto body se nazývají support vectors (podpůrné vektory) přávě proto, že jejich funkcí je podpora oddělovací roviny, která na nich spočívá.

3.7.6 Kovarianční matice

Základními charakteristikami vícerozměrného rozdělení je vektor středních hodnot

$$E(X) = \begin{bmatrix} E(X_1) \\ E(X_2) \\ \vdots \\ E(X_p) \end{bmatrix} \tag{3.26}$$



Obrázek 3.9: Princip lineárního oddělení dvou tříd s nelineárními hranicemi pomocí přidané dimenze

a kovarianční (varianční) matice

$$\Sigma = var(X) = cov(X) = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2}^2 & \cdots & \sigma_p^2 \end{bmatrix} \tag{3.27}$$

kde σ_{ij} je kovariance dvou náhodných veličin, tj.

$$\sigma_{ij} = cov(X_i, X_j) = E(X_i - EX_i)(X_j - EX_j) \tag{3.28}$$

a $\sigma_{ii} = \sigma_i^2$ je rozptyl $var(X_i)$. Vidíme, že kovarianční matice Σ je symetrická, neboť $\sigma_{ij} = \sigma_{ji}$.

3.7.6.1 Vícerozměrné normální rozdělení

Náhodný vektor X má vícerozměrné rozdělení, jestliže jeho hustota je dána vztahem

$$f(x) = (2\pi)^{\frac{p}{2}} \quad |\Sigma|^{-\frac{1}{2}} \quad exp(-\frac{(x-\mu)^T\Sigma^{-1}(x-\mu)}{2}) \tag{3.29}$$

kde μ je vektor středních hodnot a Σ je kovarianční matice.

Vícerozměrné rozdělení má tyto vlastnosti:

- lineární kombinace prvků z X mají normální rozdělení
- všechny podmnožiny X mají normální rozdělení
- nekorelovanost veličin z X znamená i jejich nezávislost
- všechna podmíněná rozdělení jsou normální

Pro jednorozměrné normální rozdělení z předchozí rovnice dostaneme

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(x-\mu)^2}{2\sigma^2}) \tag{3.30}$$

V exponenetu je čtverec vzdálenosti

$$u^2 = (\frac{x-\mu}{\sigma})^2 \tag{3.31}$$

tedy vzdálenosti x od střední hodnoty μ, kde jednotkou vzdálenosi je σ. I pro vícerozměrné normální rozdělení je možno chápat kvadratickou formu v exponentu jako čtverec vzdálenosti vektoru x od vektoru μ, ve kterém je obsažena i informace z kovarianční matice

$$C^2 = (x-\mu)^T\Sigma^{-1}(x-\mu) \tag{3.32}$$

C je Mahalanobisova vzdálenost, pro zvolenou hodnotu $f(x)$ její čtverec je geometricky plocha elipsoidu se středem μ a osami $c\sqrt{\lambda_j v_j}$ pro $j = 1, 2, \ldots, p$, kde λ_j jsou vlastní čísla matice Σ a v_j jsou vlastní vektory matice Σ [10].

$$C^2 = (X-\mu)^T\Sigma^{-1}(X-\mu) \tag{3.33}$$

3.7.7 Bi-clustering

Nechť M je mmnožina vzorků a N množina příznaků. Dále je dána matice $D = (a_{ij})_{M\times N}$ kde a_{ij} je hodnota i-tého vzorku a j-tého vzorku. Označíme-li indexy řádků a sloupců matice $D_{M\times N}$ jako $R = \{1, 2, \ldots, M\}$ a $C = \{1, 2, \ldots, N\}$ dostáváme $D = (R, C) \in \Re^{M\times N}$. Obecně biklastrování je podmnožina takových řádků, které vykazují podobné vlastnosti skrz podmnožinu sloupců. Biklastr $B = (X, Y)$ je podmatice matice D s některými podobnými vzory, kde $X = \{M_1, .., M_x\} \subset R$ a $Y = \{N_1, \ldots, N_y\} \subset C$ jsou oddělené podmnožiny R a CC. Cílem biklastrování je objevit množinu biklastrů $B_k = (K_k, Y_k)$ takových, že každý biklastr odpovídá nějaké specifické charakteristice homogenity.

Na obrázku č. 3.10 je na podobrázku (a) matice o velikosti 6×6 se skrytými biklastry. Podobrázek (b) ukazuje biklastr s konstatními hodnotami, na podobrázku (c) je vybraný biklastr hodnot s konstantními hodnotami v řádcích, (d) má konstatní sloupce. Podobrázek (e) je biklastr přírůstků v jednotlivých sloupcích. Biklastr na podobrázku (f) znázorňuje multiplikativní model.

	F_1	F_2	F_3	F_4	F_5	F_6
O_1	10	10	10	14	7	10
O_2	10	14	10	24	35	10
O_3	15	15	15	12	20	35
O_4	20	20	20	17	25	40
O_5	25	25	25	22	30	45
O_6	10	14	30	27	35	50

(a)

	F_1	F_3	F_6
O_1	10	10	10
O_2	10	10	10

(b)

	F_1	F_2	F_3
O_1	10	10	10
O_3	15	15	15
O_4	20	20	20
O_5	25	25	25

(c)

	F_1	F_2	F_5
O_2	10	14	35
O_6	10	14	35

(d)

	F_3	F_4	F_5	F_6
O_3	15	12	20	35
O_4	20	17	25	40
O_5	25	22	30	45
O_6	30	27	35	50

(e)

	F_5	F_6
O_1	7	10
O_6	35	50

(f)

Obrázek 3.10: Příklad biklastrovací metody

3.7.7.1 Biklastrovací algoritmus

Biklastering založený na výpočtu vzdálenosti (distance-based biclustering) nejčastěji používá nějakou metriku založenou na měření vzdálenosti kvality biklastrů a snaží se iterativně hledat minimalizací sum kvadrátů cen. Následující měření pomocí sum kvadrátů je použito pro vyhodnocení kvality každého biklastru $B_k = (X_k, Y_k)$:

$$SSQ_k = \sum_{i \in X_k, j \in Y_k} (a_{ij} - a_{X_k Y_k})^2 \tag{3.34}$$

kde a_{ij} je průměrná hodnota v biklastru B_k. Biklastry s nízkou hodnotou SSQ jsou považovány za lepší než ty s vyšší hodnotou SSQ. V přímém klastrování počet biklastru je fixní a řešení je dosaženo minimalizováním sumy SSQ_k.

V Cheng a Church δ-bicluster algoritmu je biklastrování založeno na minimalizaci průměru čtvercového rezidua:

$$H(X, Y) = \frac{1}{|X||Y|} \sum_{i \in X, j \in Y} (a_{ij} - a_{iY} - a_X j + a_{XY})^2 \tag{3.35}$$

kde a_{iY}, a_{Xj}, a_{XY} jsou průměr hodnot řádku, průměr hodnot sloupce a průměr v podmatici $B = (X, Y)$. Biklastr se nazývá δ-biklastr pokud $H(X, Y) \leq \delta$ pro nějaké $\delta > 0$. Abychom našli δ-biklastr, tak je nutné vypočítat ohodnocení H pro každé možné přidání nebo odebrání řádku nebo sloupce. Akce, která sníží ohodnocení H, je aplikována. Bicluster je nalezen, když už nelze snížit ohodnocení H nebo když $H \leq \delta$. Po nalezení δ-biklastru jsou elementy v odpovídající podmatici nahrazeny náhodnými čísly před tím, než se začne hledat další δ-biklastr. δ-biklastry jsou úspěšně nalezeny v původní matici, pokud se dosáhne nalezení předem specifikovaného počtu biklastrů [8].

3.7.8 k-tý nejbližší soused

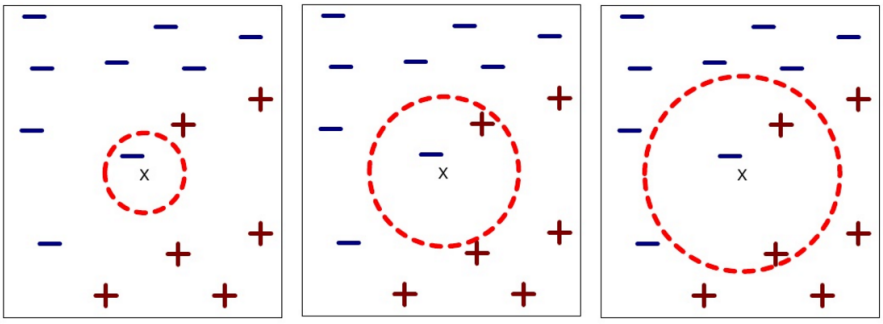
Klusterizace podle nejbližších sousedů (kth nearest neighbours) spadá mezi neparametrické metody klasifikace. Myšlenka tohoto algoritmu je založena na nalezení určitého počtu pozorování k, která jsou nejblíže hledanému bodu. Předpokládejme že $X \in \mathbb{R}^q$ a máme následující vzorky $\{X_1, \ldots, X_n\}$. Pro každý fixní bod $x \in \mathbb{R}^q$ můžeme spočítat, jak je daleko od každého vzorku X_i je x pomocí Euklideanovy vzdálenosti $\|x\| = (x'x)^{\frac{1}{2}}$. Tato vzdálenost je dána rovnicí

$$D_i = \|x - X_i\| = ((x - X_i)'(x - X_i))^{\frac{1}{2}} \tag{3.36}$$

Jedná se o jednoduchý výpočet nad množinou vzorků. Po výpočtu vzdáleností dostáváme množinu D vzdáleností od vzorků X_i. Poté tuto množinu vzestupně seřadíme. Nyní první prvek seřazené množiny D odpovídá nejbližšímu sousedovi fixního bodu x, druhý prvek seřazené množiny D odpovídá druhému nejbližšímu sousedovi atd. Tyto vzdálenosti nám dávají ohodnocení, jak jsou vzorky X_i od fixního bodu x. Můžeme si představit, že kolem bodu x budeme kreslit kružnici, která se bude postupně zvětšovat. Jakmile kružnice dosáhne některého vzorku X_i, tak tento vzorek označíme jako první nejbližší soused fixního bodu x. Jak se kružnice bude zvětšovat dál a dosáhne dalšího vzorku, tak tento vzorek je druhý nejbližší soused. Názorná ukázka je na obrázku č. 3.11. Jak se kružnice zvětšuje, tak nakonec dosáhne všech vzorků v pořadí podle nejbližších sousedů $\{X_{(1)}, X_{(2)}, \ldots, X_{(n)}\}$, k-tý nejbližší soused fixního bodu x je $X_(k)$. Pro dané k nechť

$$R_x = \|X_{(k)} - x\| = D_k \tag{3.37}$$

značí Euklidovu vzdálenost mezi x a $X_{(k)}$. R_x označuje výpočet pro k-té pořadí v množině vzdáleností D. [7]



Obrázek 3.11: Hledání nejbližšího souseda

Kapitola 4

Aplikace grafické vizualizace

Společnost Safibra s.r.o. ke svému projektu FiberSence doposud používá k analýze naměřených událostí katalog ve formátu PDF, který čítá až několik set stránek. Hledání a studium jednotlivých událostí v tomto katalogu je dost zdlouhavé a nepřehledné. Porovnávání podobností jednotlivých událostí je skoro nemožné, protože se neporovnávají digitální cestou.

Proto vznikl požadavek společnosti Safibra s.r.o. na aplikaci, která by usnadnila prohlížení a efektivně vizualizovala naměřená data. Tento požadavek byl již řešen v předmětu zabývajícím se softwarovým projektem (`A4M36SVP`), kde vznikl prototyp aplikace, která umožňuje zobrazení naměřeného signálu z provedených experimentů.

Jak již bylo zmíněno v kapitole 2.4 jsou data předem předzpracovávána aplikací různých filtrů na naměřený signál z optické senzorové sítě. Tato data jsou i vstupními daty této práce a byly na nich prováděny experimenty.

Vstupní data do aplikace jsou uložena v souboru ve formátu HDF (podrobná struktura je popsána v kapitole 4.2.1).

4.1 Specifikace požadavků

Cílem bylo analyzovat, navrhnout a naimplementovat prototyp aplikace pro společnost Safibra s.r.o., která umožní uživateli prohlížet naměřená data z provedených experimentů. Jedná se tedy o proof of concept, že implementace takové aplikace splní očekávané požadavky a pomůže s náhledem a vyhodnocováním naměřených dat. Protože se jedná o vývoj prototypu, tak byl zvolen typ prototypového přístupu vývoje aplikace. Interval vývoje nových funkcionalit do prototypu byl dohodnut na 1 týden. Takže během vývoje byly vytvářeny každý týden neúplné verze aplikace (prototypy), které následně byly prezentovány a diskutovány, zda některou z funkcionalit rozšířit nebo vynechat.

4.1.1 Model požadavků

Model požadavků vznikal iterativně během vývoje. Následující popis je souhrnný seznam požadavků, které byly kladeny na aplikaci během vytváření prototypů.

- `Návrh grafického rozhraní aplikace`
  Součástí tohoto požadavku bylo nutno se seznámit se strukturou vstupních dat. Na základě této struktury navrhnout rozložení grafického rozhraní a princip práce s projektem.

- `Zobrazení naměřeného signálu`
  Aplikace bude umožňovat zobrazení konkrétního naměřeného signálu z experimentu. Zobrazovaný signál bude zobrazen celý (od začátku experimentu až do konce).

- `Možnost manipulace v okně se signálem`
  Na základě požadavku zobrazení signálu je nutné umožnit uživateli možnost manipulace a navigace v prohlíženém signálu operacemi:
  - `zoom` - možnost přibližování a oddalování
  - `pan` - možnost navigace v signálu pomocí stisku a tažení kursoru myši
  - `percentil` - možnost omezení extrémů v prohlíženém signálu
  - `zoomhelper` - umožní automatické přibližování a oddalování signálu v prohlíženém časovém intervalu

- `Zobrazení více signálů v jednom okně`
  Možnost zobrazit více signálů v jednom okně. Aplikace bude schopna zobrazit výstupy z výstupů Fourierovy transformace a Wigner-Ville distribuce.

- `Zobrazení seznamu detekovaných anomálií`
  Aplikace zobrazí seznam detekovaných anomálií. Zároveň nabídne možnost procházení, zobrazení a zvýraznění jednotlivých detekovaných anomálií.

- `Zobrazení zachycených obrázků během experimentu`
  Aplikace zobrazí v samostatném okně obrázek, který odpovídá aktuálně prohlížené anomálii v signálu.

- `Zobrazení zachyceného videa`
  Přidání možnosti importování a zobrazení zaznamenaného videa z experimentu. K tomuto požadavku se váží další požadavky na funkcionalitu:
  - `krokování videa` - video lze prohlížet po snímcích
  - `korekce času` - nastavení zpoždění videa vůči signálu

- `Synchronizace jednotlivých zobrazených signálů`
  Aplikace bude zobrazovat signály v jednotlivých oknech synchronizovaně vůči času. Aby bylo možné posouvat signál v jednotlivých oknech automaticky najednou.

- `Možnost exportu dat, obrázků`
  Uživatel bude mít možnost exportovat prohlížený signál do obrázkového souboru nebo textového souboru s právě zobrazovanými hodnotami.

- `Konfigurovatelnost projektu`
  Aplikace bude umožňovat konfigurovatelnost následujících komponent:

- `grafické rozhranní` - uživatel má možnost určit, jaké komponenty chce zobrazit
- `konstanty` - uživatel může nastavovat používané konstanty v aplikaci

4.2 Analýza a návrh řešení

V této kapitole je v první části popsána analýza jednotlivých požadavků aplikace a druhá část se zabývá řešením aplikace po technické stránce.

4.2.1 HDF struktura

Aplikace bude používat HDF soubor jako vstupní data. Tento soubor obsahuje strukturovaná data, která jsou rozdělena do následujících pro nás důležitých sekcí:

- **Configuration** - konstanty použité při zpracování dat
- **Events** - detekované události v signálu
- **Preprocessing** - transformovaná data
- **RawData** - data ze senzoru

4.2.2 Analýza požadavků

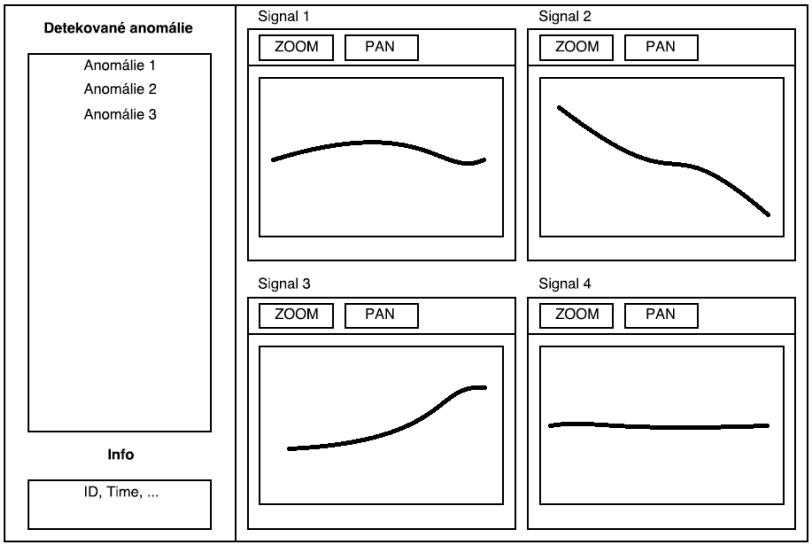
Zde si představíme analýzu důležitých požadavků a provedených kroků při jejich navrhování. Jak již bylo zmíněno na začátku kapitoly, tato aplikace je prototypem, proto se analýza požadavků skládá z nasbíraných požadavků během celého vývoje aplikace.

4.2.2.1 Návrh grafického rozhraní aplikace

Tento požadavek bylo potřeba analyzovat po celou dobu vývoje aplikace. Vzhledem k přibývajícím funkcionalitám bylo potřeba přizpůsobovat grafické rozhraní, aby práce s aplikací byla ve výsledku intuitivní a pro uživatele přehledná. Na obrázku č. 4.1 je znázorněn návrh grafického rozhraní, jak byl navržen na začátku. Vlevo v návrhu je seznam detekovaných anomálií a vpravo jsou do mřížky vyskládána okna s jednotlivými signály. Tento návrh byl v průběhu vývoje rozšířen o horní menu a toolbary u jednotlivých oken. V rámci analýzy bylo grafické rozhraní navrženo do plovoucích oken, aby bylo možné s nimi kdykoliv manipulovat dle potřeby.

4.2.2.2 Synchronizace jednotlivých zobrazených signálů

Požadavek na aplikaci vychází z předpokladu, že všechna okna zobrazující signál mají společnou časovou osu. Nejprve byla navigace v signálu z experimentu zcela v rukou uživatele, který si posouval jednotlivými okny podle potřeby. V průběhu používání se ukázalo, že mnohem výhodnější a pohodlnější by bylo, kdyby jednotlivá okna se signály se posouvala současně. Zpřehlední to prohlížení signálů a nenutí uživatele stále posouvat všechna okna na požadovaný čas. Ke konci vývoje se již zobrazuje u jednoho experimentu až 16 oken, tudíž by bylo nemyslitelné, aby byl uživatel nucen obsluhovat všechna okna. Realizace je popsána v části implementace.



Obrázek 4.1: Návrh grafického rozhraní

4.2.2.3 Konfigurovatelnost projektu

S přibývajícími funkcionalitami a úpravami pro jednotlivé experimenty se zvýšila potřeba měnit různé konstanty a konfigurace jednotlivých komponent. Proto se většina konfigurace během vývoje přemístila do konfiguračního souboru.

4.2.3 Návrh řešení

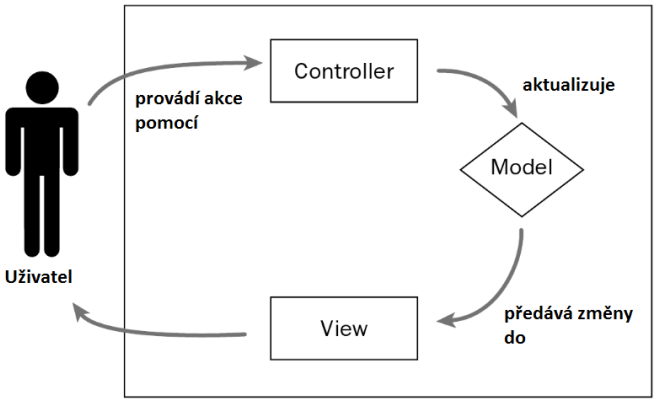
Protože se jedná o vývoj prototypu aplikace. zvolil jsem programovací jazyk Python, který nabízí rychlé psaní a testování kódu a velké množství implementovaných modulů, které je možné okamžitě použít v aplikaci. Zvolil jsem vývojové prostředí Anaconda, které zahrnuje vše potřebné pro vývoj aplikace v jayzce Python, včetně přehledného IDE Spyder. Výchozí verze jazyka Python je 2.7, která obsahuje implementaci všech použitých knihoven v naší aplikaci. Například se jedná o knihovnu pro práci s HDF soubory nebo knihovna PyQt4 pro grafické rozhraní aplikace a knihovnu matplotlib, která umožňuje vykreslovat signály, které se následně zobrazují v oknech aplikace.

4.2.3.1 Python

Python je dynamický interpretovaný jazyk. Někdy bývá zařazován mezi takzvané skriptovací jazyky. Jeho možnosti jsou ale větší. Python byl navržen tak, aby umožňoval tvorbu rozsáhlých, plnohodnotných aplikací. Python je hybridní jazyk (nebo také víceparadigmatický), to znamená, že umožňuje při psaní programů používat nejen objektově orientované paradigma, ale i procedurální a v omezené míře i funkcionální, podle toho, co se pro danou úlohu hodí nejlépe. Díky tomu má Python vynikající vyjadřovací schopnosti. Kód programu je ve srovnání s jinými jazyky krátký a dobře čitelný.

4.2.3.2 Struktura aplikace

Aplikace je rozdělelna do jednotlivých vrstev, které vycházejí z architektonického návrhu MVC (model view controller). Obecně se dá proces používání aplikace popsat obrázkem č. 4.2. Uživatel komunikuje s aplikací pomocí uživatelského rozhraní, které snímá akce uživatele. Na základě akce uživatele zareaguje model (aktualizují se data) a následně i grafické rozhraní.



Obrázek 4.2: MVC architektonický vzor

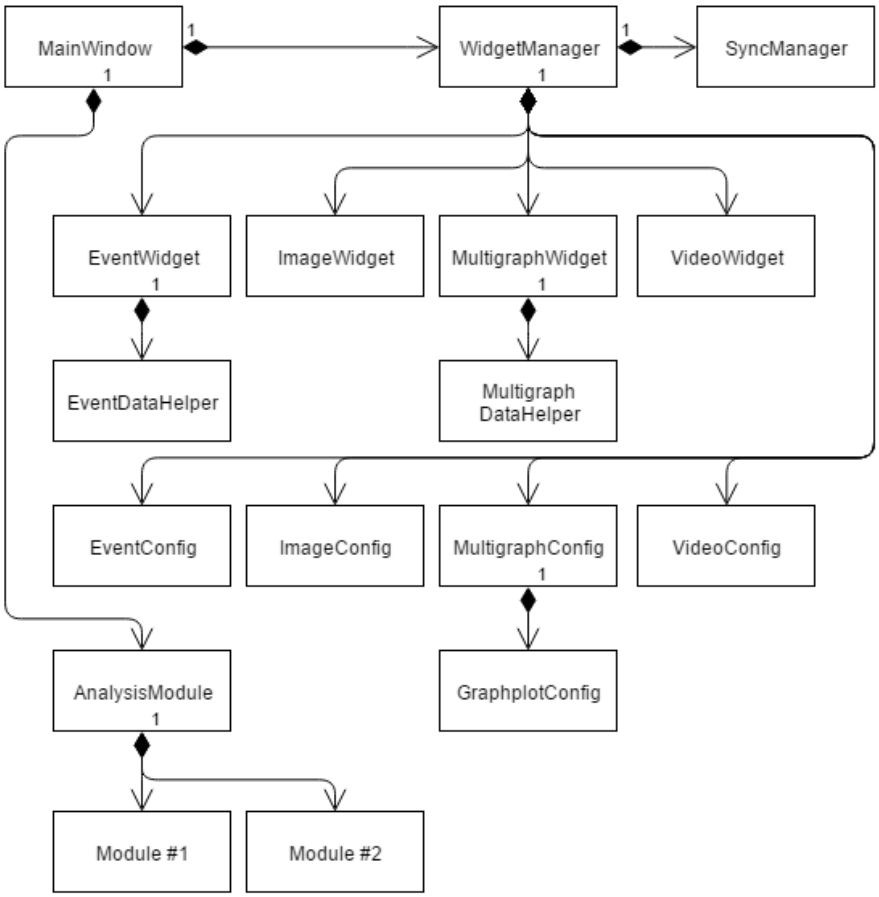
4.2.4 Diagram tříd

Diagram tříd aplikace je zobrazen na obrázku 4.3, popisuje strukturu a vazby mezi jednotlivými objekty. Zde jsou třídy dle sufixu názvu rozděleny podle jejich funkcionality:

- `Window, Widget` - implementace grafické komponenty
- `DataHelper` - implementace pro práci s daty
- `Manager` - implementace funkcionality aplikace
- `Config` - implementace reprezentující konfiguraci objektu
- `Module` - implementace analytických funkcí nad daty

4.3 Realizace

Tato kapitola se zabývá popisem klíčových bodů implemetace aplikace, které je možné použít pro reimplementaci této aplikace do jiného programovacího jazyka. Implementace prototypu aplikace se během postupného vývoje měnila a upravovala dle potřeb požadavků. V rámci vývoje se pokračovalo od primitivní aplikace až k současnému stavu aplikace. Během této doby prošel prototyp několika desítkami úprav, vylepšení a optimalizací. V dalších kapitolách si popíšeme postup vývoje důležitých částí aplikace.



Obrázek 4.3: Model tříd

4.3.1 Widget

Widget je grafická komponenta, která je schopná reprezentovat data uživateli v grafické podobě. Implementace vychází z třídy `DockWidget` grafické knihovny PyQt4. Tento druh widgetu je možné dokovat v rámci aplikace nebo ho libovovlně umisťovat i mimo okno aplikace. Aplikace implementuje celkem čtyři různé widgety pro reprezentaci dat. Konkrétně to jsou

- `MultigraphWidget` - zobrazuje průběhy naměřených signálů v čase
- `EventWidget` - zobrazuje seznam detekovaných anomálií v signálu
- `ImageWidget` - spolupracuje s `EventWidget` a zobrazuje uložené obrázky pro jednotlivé anomálie
- `VideoWidget` - zobrazuje videosnímky videosouboru v čase prohlíženého intervalu

4.3.2 Konfigurace widgetů

Jednotlivé widgety jsou v aplikaci konfigurovány dynamicky. Uživatel má možnost si sestavit konfigurační XML soubor (definice XML souboru je popsána v příloze C.1). Aplikace dává

uživateli na výběr, jaké widgety chce nakonfigurovat a následně zobrazit:

- <**graphwidget**> - vytvoří `MultigraphWidget`
- <**eventwidget**> - vytvoří `EventWidget`
- <**videovideget**> - vytvoří `VideoWidget`
- <**imagewidget**> - vytvoří `ImageWidget`

Kromě konfigurací widgetu se v XML souboru nachází i konfigurace globálních proměnných pro projekt.

4.3.3 Synchronizace widgetů

Aplikace používá pro synchronizaci jednotlivých widgetů objekt `SynchronizationManager`. Jednotlivé widgety se po spuštění aplikace přihlásí k tomuto manageru. Předpokladem zařazení widgetu do synchronizace je, aby data v tomto widgetu byla časově svázána s ostatními widgety. Uživatel má možnost kdykoliv odebrat nebo přidat jakýkoliv widget z a nebo do synchronizace.

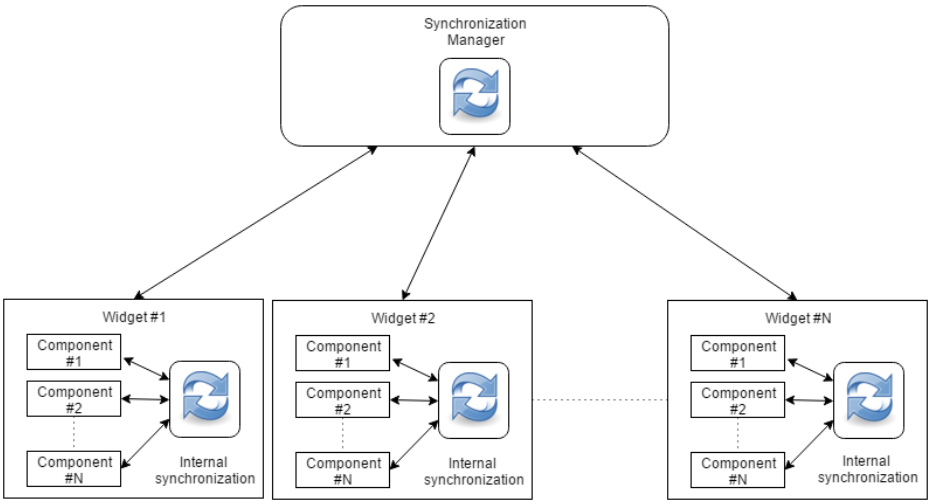
Úkolem synchronizačního managera je udržovat konzistentní pohled na data ve widgetech.

Funkce synchronizačního procesu je znázorněná na obrázku č. 4.4. Každý widget má implementovanou interní synchronizaci jednotlivých komponent, která zajišťuje přenastavení do korektního stavu. Například pokud uživatel při prohlížení signálu použije scrollbar, pak je nutné synchronizovat plátno a vykreslit část signálu, která odpovídá nové hodnotě. Po provedení interní synchronizace tento widget odešle objektu `SynchronizationManager` zprávu s novým stavem. Synchronizační manager odešle notifikaci ostatním widgetům, krom widgetu, který tuto zprávu poslal (funkčnost lze připodobnit k funkčnosti síťového prvku HUB). Ostatní widgety přijmou tuto notifikaci s údaji, do jakého stavu se mají nastavit. Každý druh widgetu obsahuje implementaci zpracování notifikací od ostatních druhů widgetů.

Může nastat i situace, kdy widget se nachází v nekonzistentním stavu (není synchronizován) a je potřeba ho zařadit zpátky do synchronizace. V tuto chvíli i po přihlášení do synchronizace, se stav widgetu nezmění. Změna nastane, až když proběhne další synchronizační proces vyvolaný změnou od uživatele. Tento krok je zde proto, že nevíme podle jakého widgetu bude chtít uživatel sesynchronizovat projekt.

4.3.4 Widget manager

Implementace `WidgetManager` je důležitá pro grafické rozhraní. Obstarává načtení konfiguračního souboru XML, které v sobě obsahuje uživatelovo nastavení. Tento soubor je parsován a dle obsahu jsou vytvořeny instance konfigurací pro jednotlivé komponenty. Když máme načteny všechny instance konfigurací widgetů, tak se následně podle nich sestaví instance jednotlivých widgetů. Instance widgetů jsou uloženy do patřičných seznamů. Ve chvíli, kdy je potřeba, si instance `MainWindow` vezme tyto instance a vytvoří z nich dokovací widgety a umístí je do hlavního okna aplikace.



Obrázek 4.4: Synchronizace widgetů - schéma

4.3.5 Zoom helper

Zoom heplper slouží k automatickému přiblížení či oddálení prohlíženého signálu. Tato utilita umožňuje plynulé prohlížení signálů, které vykazují klesající nebo stoupající trend. Uživatel může určit, zda pro danou instanci `MultigraphWidget` chce tuto funkci zapnout či vypnout. Funkcionalita je založená na rychlém vyhledání zobrazeného intervalu dat a nad nimi spočítá maximální a minimální hodnoty. Poté podle těchto hodnot upraví přiblížení či oddálení signálu.

4.4 Testování a optimalizace

Testování aplikace probíhalo kontinuálně během vývoje a používání aplikace. Protože se jedná o vývoj prototypu aplikace, tak během vývoje byla testována pouze hlavní funkcionalita aplikace. Nebyly testovány všechny případy užití ve všech vyvinutých verzích aplikace. Ve chvíli, kdy se vyskytla chyba, tak v další verzi aplikace došlo k její nápravě. Během vývoje došlo ke změně generování HDF souboru. Po načtení nové struktury se projevila chyba v event widgetu, který skončil chybou při zobrazování anomálie, protože algoritmus pro vyhledávání měl nastavenou špatnou podmínku terminace. V další verzi již byla tato chyba opravena.

S přibývající funkcionalitou aplikace bylo potřeba dát důraz i na optimalizaci používaných algoritmů. Z počátku vývoje bylo možné psát funkcionalitu a nezabývat se náročností použitých algoritmů. Ve chvíli, kdy přibyl požadavek pro synchronizaci widgetů bylo potřeba optimalizovat rendrování zobrazovaných signálů. Další nutná optimalizace kódu proběhla při implementaci požadavku pro automatické zoomování signálu, kde bylo potřeba zrychlit vyhledávání extrémů v intervalu zobrazovaného signálu. Použití této funkce v experimentech trvajících několik desítek minut byla tato utilita nepoužitelná.

Aplikace byla testována i v prostředí Anaconda Python 3.5 64-bit, kde vykazovala zpoždění při rendrování widgetů. Proto je doporučováno používat Python 2.7, kde se aplikace chová stabilně.

Aplikace byla úspěšně testována i na 4K displeji, kde je aplikace bez problému ovladatelná.

4.5 Nasazení

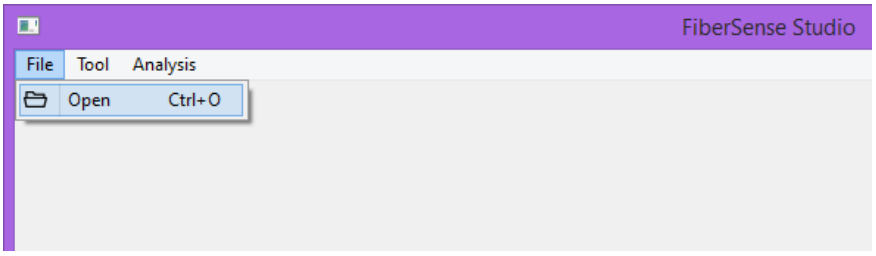
Prototyp aplikace je implementovaný na osobním počítači, kde je schopný sloužit k prohlížení naměřených experimentů. V prototypu aplikace jsou implementovány moduly pro analýzu a rozpoznávání detekovaných anomálií. Aktuální protoyp aplikace není optimalizován, aby efektivně využíval služby clusteru. Při reimplementaci aplikace je potřeba vzít v úvahu paralelizaci algoritmů v jednotlivých analyzačních modulech. Pak je možné aplikaci spustit na clusteru, kde se analýza může provádět na více strojích najednou a tím zrychlit vyhodnocení zkoumané anomálie.

4.6 Uživatelský manuál

V této kapitole je představena základní funkcionalita aplikace a základní případy užití aplikace.

4.6.1 Otevření HDF souboru

FiberSense Studio je grafická aplikace pro vizualizaci naměřených signálů z experimentu využívjící grafické komponenty z knihovny PyQt4. Po spuštění programu má uživatel možnost vybrat z menu `File -> Open` pro otevření HDF souboru s předzpracovanými daty (viz. obrázek č.4.5).

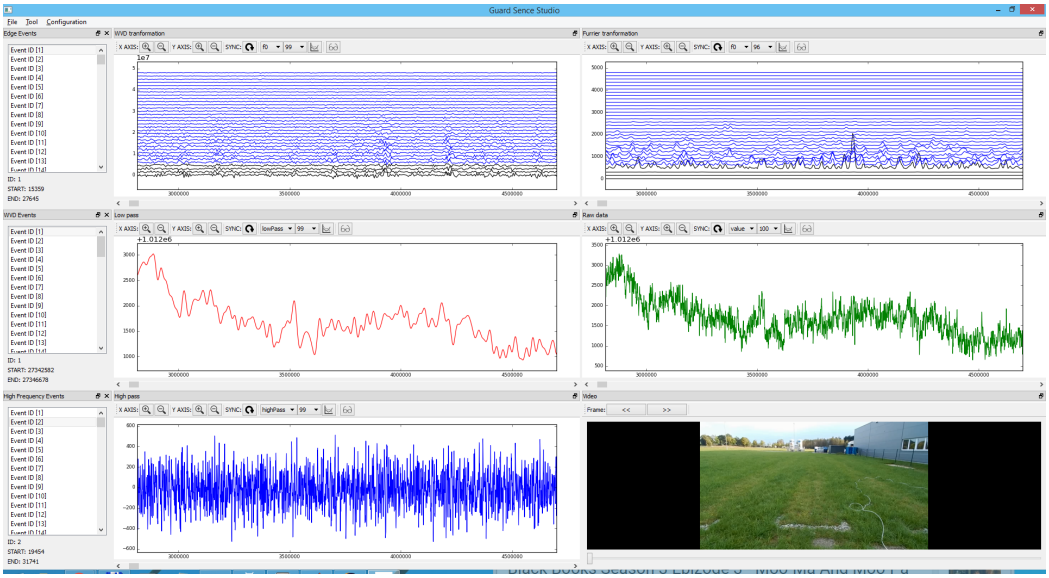


Obrázek 4.5: Otevření souboru

Po vybrání HDF souboru se načte defaultní konfigurace ze souboru XML, kde jsou definovány jednotlivé widgety, které se mají zobrazit a jejich parametry (tabulka s daty, jména sloupců atd. . . .). Více o konfiguraci aplikace je popsáno v kapitole 4.3.2.

4.6.2 Popis grafického prostředí

Po načtení HDF souboru se aplikace nakonfiguruje dle konfigurace uvedené v XML souboru a načte se grafické prostředí. Ukázka je zobrazena na obrázku č. 4.6. Vlevo jsou načteny seznamy detekovaných anomálií a vpravo jednotlivé widgety.



Obrázek 4.6: Grafické prostředí aplikace

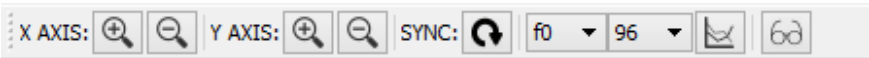
4.6.2.1 Graph widget

Graph widget slouží k zobrazení jednotlivých signálů na časové ose. Je možné definovat zobrazení jednoho nebo více průběhů v rámci jednoho widgetu. V rámci widgetu je implementovaný `toolbar` pro ovládání jednotlivých akcí. Pohyb po časové ose je možné díky scrollbaru umístěném dole ve widgetu nebo pomocí stisknutím levého tlačítka myši a tažením.

`Toolbar`
Pomocí toolbaru je možné provádět akce

- `Zoom` - Zoomování na x-ové nebo y-ové ose. Zoomování je zarovnáno k levému okraji okna.
- `Synchronizace` - Zakázat/povolit synchronizaci s ostatními widgety.
- `Percentil` - Přenastavení hodnoty percentil pro jednotlivé průběhy. Po potvrzení změny dojde k překreslení widgetu a nastavení do původní polohy, ve které se nacházel před změnou hodnoty percentil.
- `Zoomhelper` - Zakázat/povolit utilitu pro automatické zoomování zobrazováného signálu.

Změny jsou pouze lokální a neovlivní hodnoty v konfiguračním souboru XML.



Obrázek 4.7: Toolbar pro graph widget

4.6.2.2 Image widget

Image widget zobrazuje obrázek pro aktuálně prohlíženou anomálii. Ke změně obrázku dojde buďto po výběru konkrétní anomálie ze seznamu vlevo nebo při posouvání se v časové ose. Pokud se prohlížený interval signálu v grafickém widgetu nachází v intervalu nějaké anomálie, zobrazí se obrázek uložený pro tuto konkrétní anomálii.

4.6.2.3 Event widget

Event widget zobrazuje seznam detekovaných anomálií. Uživatel zde může vybrat jednotlivou anomálii a zobrazit si její průběh v ostatních widgetech. Po dvojkliku dojde k nastavení všech widgetů do pozice začátku a konce anomálie. Při pohybu po časové ose v jiných widgetech se aktualizuje pozice aktuální prohlížené anomálie. Widget má stejně jako grafický widget implementovaný `toolbar`.

`Toolbar`
Toolbar, který je zobrazen na obrázku č. 4.8 umožňuje uživateli barevně označit/odznačit časové intervaly anomálií v grafických widgetech, což umožní vizuálně zvýraznit jednotlivé anomálie. Další funkcí je možnost manuální klasifikace jednotlivých anomálií. Více je popsáno v kapitole 4.6.2.5.

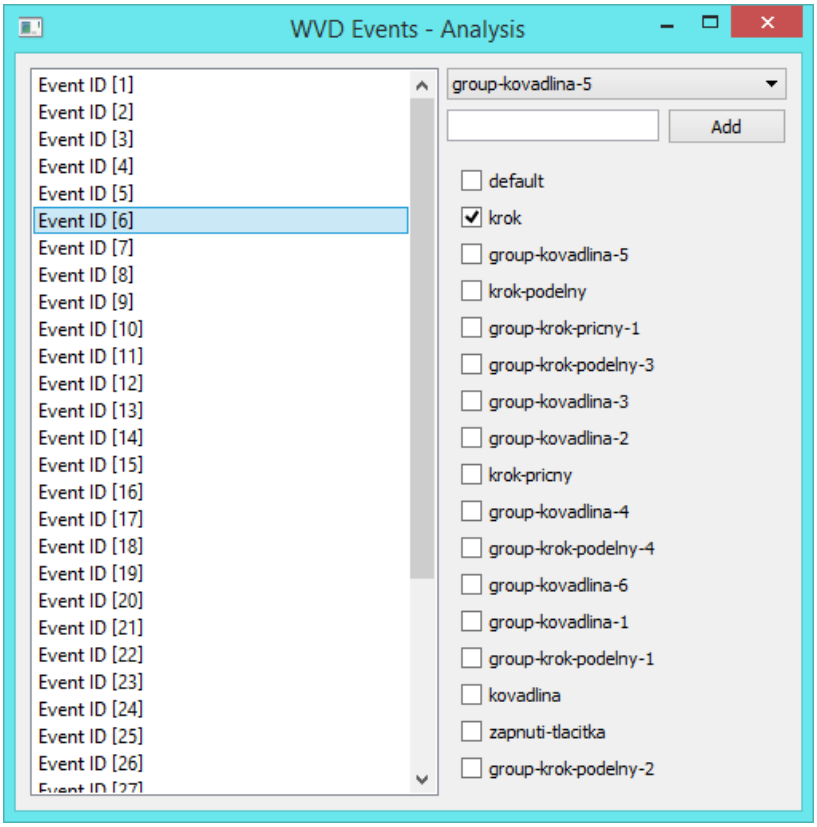
Obrázek 4.8: Toolbar pro event widget

4.6.2.4 Video widget

Video widget zobrazuje pozastavené video, které je připojeno k experimentu. Umožňuje procházet signál pomocí přechodů na předchozí/další snímek nebo pomocí slideru pro procházení videa po časové ose. Tento widget je automaticky synchronizován s ostatními widgety.

4.6.2.5 Manuální klasifikace

Formulář pro manuální klasifikaci umožňuje uživateli přiřadit třídu jednotlivým detekovaným anomáliím. Díky tomuto formuláři je možné vytvořit validační množinu dat, kterou lze použít pro kontrolu funkce jednotlivých algoritmů. Na obrázku č. 4.9 je zobrazen formulář, ve kterém je na levé straně seznam detekovaných anomálií a v pravé části lze přiřadit jednotlivé třídy, do kterých patři. Pokud třída neexistuje, tak ji lze jednoduše přidat do seznamu.



Obrázek 4.9: Manuální klasifikace anomálií

Kapitola 5

Experimenty s daty

V kapitole 3.7 jsme si představili množinu algoritmů, které budeme používat v následujících experimentech. Tato kapitola popisuje postupy jednotlivých experimentů včetně jejich implementace a získaných výsledků. Závěr kapitoly se zabývá diskusí nad získanými výsledky a postřehy, které bude možno použít pro budoucí práci. Vstupní data pro experimenty jsou množiny detekovaných anomálií v jednotlivých filtrovaných nebo transformovaných signálech. Tyto experimenty byly vybrány na základě bližšího výběru z běžně používaných algoritmů pro datamining úseků signálů. V každém experimentu je popsán postup použití a kombinace s ostatními algoritmy a prezentovány výsledky s použitými vstupními konfiguracemi.

5.1 Implementace analytických modulů do aplikace

Programovací jazyk Python nabízí implemetaci dataminingových knihoven jako např. `numpy`, `sklearn`, `pandas` a další.

V aplikaci grafické vizualizace popsané v kapitole 4 je představena implementace samotné aplikace. Součástí aplikace je implementace modulů pro provádění analýzy nad konkrétní množinou dat. Tato funkcionalita usnadňuje přehlednost klasifikace nebo shlukování. Jednotlivé výsledky datamining se dají ihned porovnat a popřípadě analyzovat manuálně a zjistit, zda daný algoritmus pracuje správně.

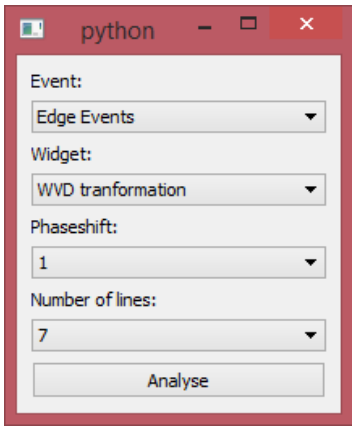
5.1.1 Analytický modul

Analytický modul obsahuje konkrétní implementaci algoritmu, který se používá pro zpracování vstupních dat a zobrazení nebo uložení dosažených výsledků.

Struktura analytického modulu je rozdělena do následujících částí:

- `init()` - přijímá vstupní parametry modulu. Seznam detekovaných anomálií, instanci `EventWidget` a `MultigraphWidget`.
- `loadsamples()` - metoda pro získání dat pro vstupní detekované anomálie
- `analyze()` - samotná implementace analyzujícího algoritmu

Pro každý analytický modul je v aplikaci konfigurační okno, kde má uživatel možnost zvolit vstupní hodnoty, které se použijí pro analýzu. Na obrázku č. 5.1 je ukázka vstupního konfiguračního formuláře, kde si uživatel může vybrat vstupní signál, vstupní množinu detekovaných anomálií a další parametry pro konfiguraci algoritmu.



Obrázek 5.1: Vstupní konfigurace analytického modulu

5.2 Testování algoritmů nad množinou dat

Kapitola je věnována popisu postupu při testování jednotlivých algoritmů. Popis jednotlivých experimentů je seřazen chronologicky tak, jak byly postupně testovány.

Jednotlivé vzorky anomálií testované v experimentech mají variabilní časovou délku a pro každý vzorek anomálie známe jeho klasifikaci z manuálního oklasifikování. K dispozici pro tuto práci byl naměřený experiment od společnosti Safibra s.r.o., kde byly testovány následující hlavní anomálie, které je potřeba rozlišit:

- `Krok` - rozlišujeme krok podélný a krok příčný (vzhledem ke směru optického vlákna)
- `Pád kovadliny` - obecně pád těžkého předmětu na zem

Experimenty obsahují úplnou specifikaci postupu, i když se text opakuje, tak každý experiment je plně určen.

5.2.1 Experiment #1

Jako první experiment provedený nad daty bylo zkoumání detekovaných anomálií pomocí algoritmu `prokládání křivek`, který je popsaný v kapitole 3.7.1.

5.2.1.1 Popis

Jednotlivé intervaly signálu jsou prokládány křivkou, která by měla vykazovat podobné charakteristické vlastnosti pro podobné nebo stejné detekované anomálie. Tato křivka se dále použije pro klasifikaci nebo shlukování.

5.2.1.2 Vstupní data

Jako vstupní data byly použité detekované anomálie ze signálů `Preprocessing`, konkrétně výstupy z filtrace vysokých a nízkých frekvencí. Pro experiment byly vybrány vzorky reprezentující jednotlivé události. Konkrétně následující vzorky:

- Krok - podélný (10 vzorků)
- Krok - příčný (10 vzorků)
- Pád kovadliny (2 vzorky)

5.2.1.3 Postup

Pro zpracování jednotlivých vzorků byla použita knihovna `numpy`. Konkrétně funkce `polyfit()`, která pomocí metody nejmenších čtverců přizpůsobí vstupní polynom stupně n na body (x, y) a vrátí vektor koeficientů polynomu s minimalizovanou kvadratickou chybou. Poté použijeme funkcí `poly1d`, která pro získané koeficienty vrátí konkrétní hodnoty.

5.2.1.4 Krok - příčný

Na obrázku č. A.1 je reprezentace vybraných vysokofrekvenčních vzorků příčného kroku. Každý vzorek je proložen křivkou, která je vypočtena z 20 bodů původního vzorku, kterými pak tato křivka prochází. Stejný postupem jsou vytvořeny křivky pro vzorky z nízkofrakvenčního průchodu na obrázku č. A.2.

V tabulce 5.2 jsou uvedeny hodnoty jednotlivých koeficientů prokládané křivky. Zde uvádím pouze hodnoty, které nejsou zanedbatelné. (pozn.: Hodnoty pro x^8 až x^{20} byly zanedbatelně malé.)

x^7	x^6	x^5	x^4	x^3	x^2	x^1
-0.0	0.0	-0.0	0.0	-1.6	83.6	-1037.9
0.0	-0.0	0.0	-0.0	1.7	-55.7	377.3
0.0	-0.0	0.0	-0.0	4.6	-198.0	1784.0
0.0	-0.0	0.0	-0.0	2.0	-92.3	1048.5
-0.0	0.0	0.0	-0.0	0.8	-56.7	1016.4
0.0	-0.0	0.0	0.0	-0.1	0.7	347.6
-0.0	0.0	-0.0	0.0	-3.1	100.1	-443.2
-0.0	0.0	-0.0	0.0	-3.1	100.0	-443.2
0.0	-0.0	0.0	-0.0	2.6	-94.8	702.8
-0.0	0.0	-0.0	0.0	-1.8	61.4	-268.7

Obrázek 5.2: Hodnoty koeficientů prokládané křivky pro vysokofrekvenční vzorky

V následující tabulce 5.3 jsou uvedeny vypočtené koeficienty prokládané křivky pro nízkofrekvenční vzorky.

x^7	x^6	x^5	x^4	x^3	x^2	x^1
0.0	-0.0	0.4	-10.7	158.6	-968.6	993905.9
-0.0	0.0	-0.2	2.7	-13.0	-61.0	988316.1
-0.0	0.0	-0.9	22.2	-293.2	1547.1	971511.8
0.0	-0.0	0.1	-2.0	39.9	-316.7	974383.0
0.0	-0.0	0.2	-3.8	27.5	-4.2	969230.0
0.0	-0.0	0.8	-18.2	203.7	-861.8	970204.4
0.0	-0.0	0.1	0.1	-35.3	441.8	963612.6
0.0	-0.0	0.1	0.1	-35.3	441.8	963612.6
-0.0	0.0	-1.2	32.8	-481.6	2891.0	945238.9
0.0	-0.0	0.2	-5.8	71.4	-335.6	946973.9

Obrázek 5.3: Hodnoty koeficientů prokládané křivky pro nízkofrekvenční vzorky

5.2.1.5 Krok - podélný

Na obrázku č. A.3 je reprezentace vysokofrekvenčních vzorků podélného kroku. Každý vzorek je proložen křivkou, která je vypočtena z 20 bodů původního vzorku, kterými pak tato křivka prochází. Stejný postupem jsou vytvořeny vektory pro nízkofrekvenční vzorky na obrázku č. A.4.

V tabulce 5.4 jsou uvedeny hodnoty jednotlivých koeficientů prokládané křivky. Zde uvádím pouze hodnoty, které nejsou zanedbatelné. (pozn.: Hodnoty pro x^8 až x^{20} byly zanedbatelně malé.)

x^7	x^6	x^5	x^4	x^3	x^2	x^1
-0.0	0.0	-0.2	2.6	-20.2	92.5	979.4
0.0	-0.0	0.2	-5.5	93.1	-679.1	1675.2
-0.0	0.0	-0.3	9.6	-145.4	890.0	-2287.2
0.0	-0.0	0.2	-3.9	50.9	-295.0	164.2
-0.0	-0.0	0.1	-2.9	58.2	-469.4	1738.2
-0.0	0.0	-0.7	14.5	-165.7	761.3	-558.4
0.0	-0.0	0.4	-7.9	84.7	-356.6	3512.7
-0.0	0.0	-0.1	2.0	-25.1	183.6	871.4
-0.0	0.0	-0.3	5.8	-66.9	354.4	782.7
-0.0	0.0	-0.4	7.5	-81.1	335.3	2329.5

Obrázek 5.4: Hodnoty koeficientů prokládané křivky pro vysokofrekvenční vzorky

V následující tabulce 5.5 jsou uvedeny vypočtené koeficienty prokládané křivky pro nízkofrekvenční vzorky.

5.2.1.6 Pád kovadliny

Na obrázku č. 5.8 je reprezentace vysokofrekvenčních vzorků pádu kovadliny. Každý vzorek je proložen křivkou, která je vypočtena z 20 bodů původního vzorku, kterými pak tato křivka

x^7	x^6	x^5	x^4	x^3	x^2	x^1
10.0	-95.1	591.8	-2242.4	4489.7	-3386.0	908777.8
4.2	-40.7	252.9	-931.0	1708.3	-923.1	891617.3
0.9	-12.9	103.7	-483.3	1113.9	-1041.0	914369.5
-4.7	46.7	-302.4	1183.6	-2457.1	2046.8	889348.4
2.4	-19.0	93.1	-244.6	224.2	244.1	917513.7
2.0	-16.8	88.0	-261.2	354.8	-139.8	891330.1
5.3	-47.5	277.2	-971.7	1747.1	-690.7	876446.5
4.6	-46.7	308.6	-1221.3	2507.0	-1871.1	893928.4
5.2	-50.9	324.5	-1244.2	2465.0	-1622.3	898969.7
2.6	-21.5	108.6	-306.1	362.2	360.6	937958.4

Obrázek 5.5: Hodnoty koeficientů prokládané křivky pro nízkofrekvenční vzorky

prochází. Stejný postupem jsou vytvořeny vektory pro nízkofrekvenční vzorky na obrázku č. 5.9.

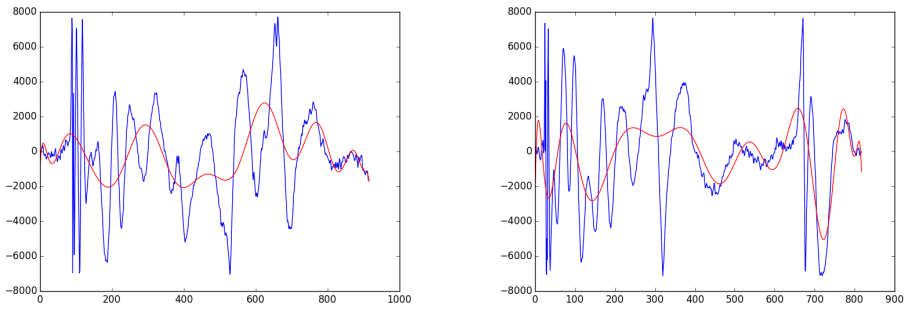
V tabulkách 5.6 a 5.7 jsou uvedeny hodnoty jednotlivých koeficientů prokládané křivky. Zde uvádím pouze hodnoty, které nejsou zanedbatelné. (pozn.: Hodnoty pro x^8 až x^{20} byly zanedbatelně malé.)

x^7	x^6	x^5	x^4	x^3	x^2	x^1
-0.0	0.0	-0.0	1.3	-36.6	394.6	-960.3
-0.0	0.0	-0.1	4.8	-125.8	1314.6	-2804.5

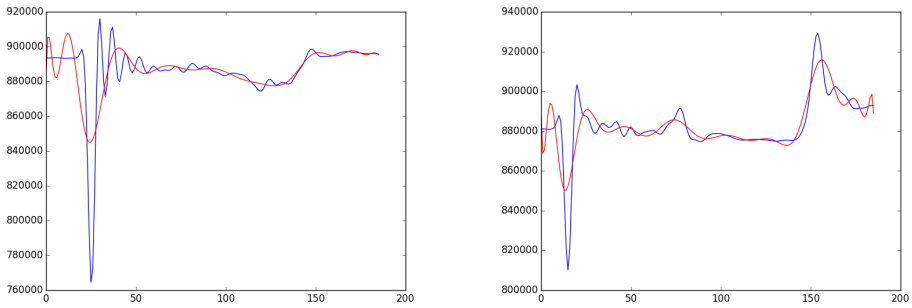
Obrázek 5.6: Hodnoty koeficientů prokládané křivky pro vzorky z tabulky bandhighpass

x^7	x^6	x^5	x^4	x^3	x^2	x^1
-2.0	43.2	-619.5	5341.1	-23972.0	42008.3	882776.9
2.3	-49.4	684.0	-5743.8	24974.8	-41501.8	890436.5

Obrázek 5.7: Hodnoty koeficientů prokládané křivky pro vzorky z tabulky bandlowpass



Obrázek 5.8: Jednotlivé vysokofrekvenční vzorky pro pád kovadliny



Obrázek 5.9: Jednotlivé nízkofrekvenční vzorky pro pád kovadliny

5.2.1.7 Vyhodnocení

Z náhledů vstupních vzorků je možné zjistit, že pouze vysokofrekvenční vzorky vykazují vzájemnou podobnost mezi sebou. U nízkofrekvenčních vzorků je vzájemná podobnost výrazně menší. Takže pro klasifikaci by bylo vhodné použít křivky proložené vysokofrekvenčními vzorky a křivky z nízkofrekvenčních vzorků buďto nepoužívat a nebo je mít pouze jako doplňující informaci, která by mohla pomoci zpřesnit klasifikaci.

5.2.2 Experiment #2

Tento experiment prováděný nad daty se zabývá zkoumáním detekovaných anomálií ve `fázovém prostoru`. Teoretický postup je popsán v kapitole 3.7.3. Pro shlukování použijeme shlukovací algoritmus `k-tý nejbližší soused`, který je popsán v kapitole 3.7.8.

5.2.2.1 Popis

Jednotlivé intervaly signálu jsou analyzovány ve fázovém prostoru. Vzniklá křivka ve fázovém prostoru by měla pro podobné anomálie vykazovat podobný tvar. Tvar této křivky následně použijeme pro klasifikaci pomocí shlukovacího algoritmu zmíněného výše.

5.2.2.2 Vstupní data

Jako vstupní data byly použité detekované anomálie ze signálů `Preprocessing`, konkrétně výstupy z filtrace vysokých a nízkých frekvencí. Pro experiment byly vybrány vzorky reprezentující jednotlivé události. Konkrétně následující vzorky:

- Krok - podélný (10 vzorků)
- Krok - příčný (10 vzorků)
- Pád kovadliny (5 vzorků)

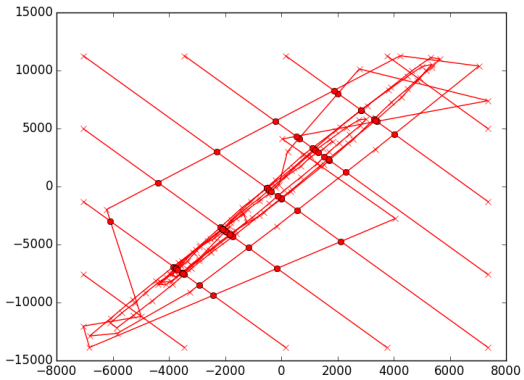
5.2.2.3 Postup

Pro jednotlivé vzorky pomocí následující funkce bylo vytvořeno dvoudimenzionální zobrazení do fázového prostoru. Budeme zkoumat fázový prostor tvořený souřadnicemi $q_{(t)} = x_{(t-5)}, x_{(t)}$, kde q je bod ve fázovém prostoru a x je hodnota signálu v čase t:

```
#array - sample values of asix y
phaseshift = 5
noise_treshold = 200
for idx, i in enumerate(array):
        if idx > phaseshift:
                if abs(i - array[idx - phaseshift]) > noise_treshold:
                        self.out_x.append(array[idx - phaseshift])
                        self.out_y.append(i + array[idx - phaseshift])
```

Výstupem funkce je zobrazení vzorku do dvoudimenzionálního prostoru, díky hodnotě `noise_treshold` se zbavíme šumových dat, která by zbytečně zkreslovala výsledky.

Nyní tento prostor proložíme přímkami, které nám budou protínat křivku v různých částí. Na obrázku č. 5.10 je znázorněn příklad proložení grafu přímkami s vyznačenými průsečíky.



Obrázek 5.10: Zobrazení ve fázovém prostoru

Příznakový vektor se skládá ze dvou získaných hodnot z fázového prostoru:

- **Maximální fázová rychlost**
- **Počet průniků s proloženými přímkami**

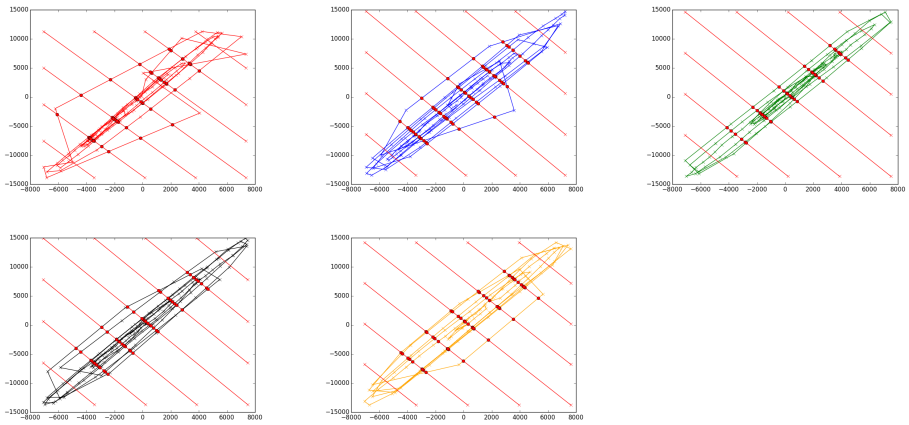
Maximální fázovou rychlost vezmeme jako první zkoumaný příznak. Druhým zkoumaným příznakem je počet průniků přímkami.

Pro jednotlivé zkoumané události budeme dostávat body v dvoudimenzionálním prostoru, které dále budeme zařazovat do shluků.

5.2.2.4 Kovadlina

Na obrázku č. 5.11 je vizualizace vzorku pádu kovadliny ve fázovém prostoru. Dále jsou zde proložené přímky a vyznačené průniky proložených přímek se vzorkem.

V tabulce č. 5.12 jsou hodnoty příznakového vektoru pro pád kovadliny.



Obrázek 5.11: Vizualizace pádu kovadliny ve fázovém prostoru

počet průniků	prům. fázová rychlost
30	3536.91434152
37	4087.50501927
26	3100.82489187
37	3769.03094554
35	4333.8089472

Obrázek 5.12: Příznakové vektory pádu kovadliny

5.2.2.5 Krok - příčný

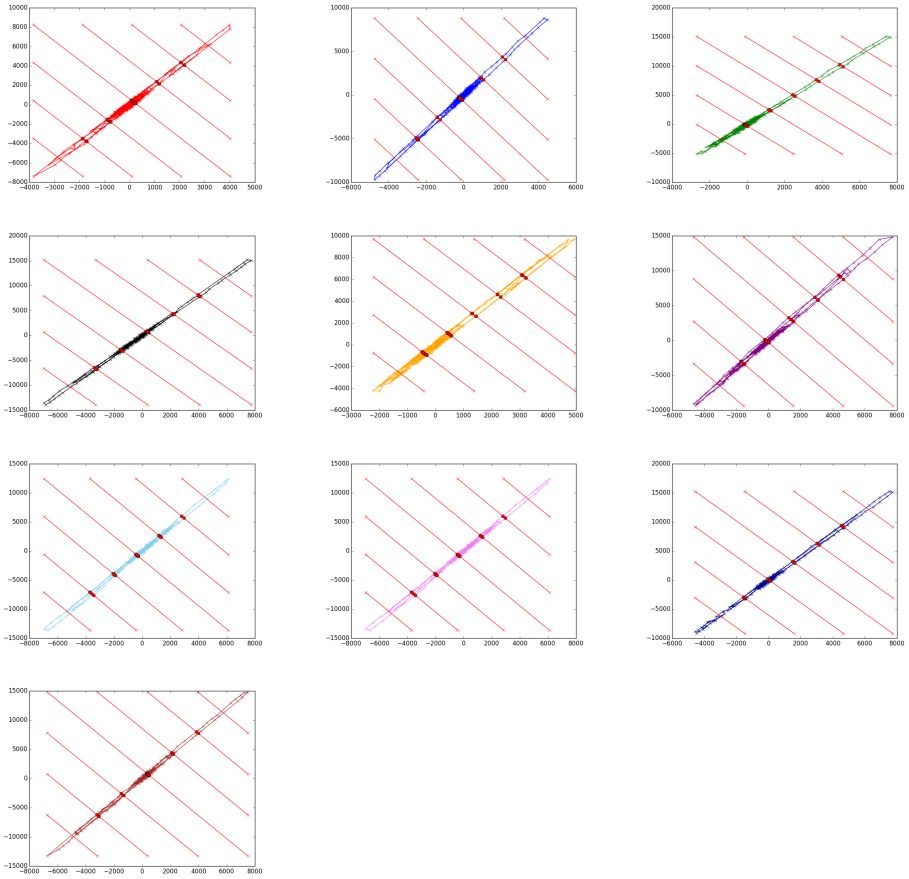
Na obrázku č. 5.13 je vizualizace vzorku příčného kroku ve fázovém prostoru. Dále jsou zde proložené přímky a vyznačené průniky proložených přímek se signálem vzorku.

V tabulce č. 5.14 jsou hodnoty příznakového vektoru pro pád kovadliny.

5.2.2.6 Krok - podélný

Na obrázku č. 5.15 je vizualizace vzorku podélného kroku ve fázovém prostoru. Dále jsou zde proložené přímky a vyznačené průniky proložených přímek se signálem vzorku.

V tabulce č. 5.16 jsou hodnoty příznakového vektoru pro pád kovadliny.



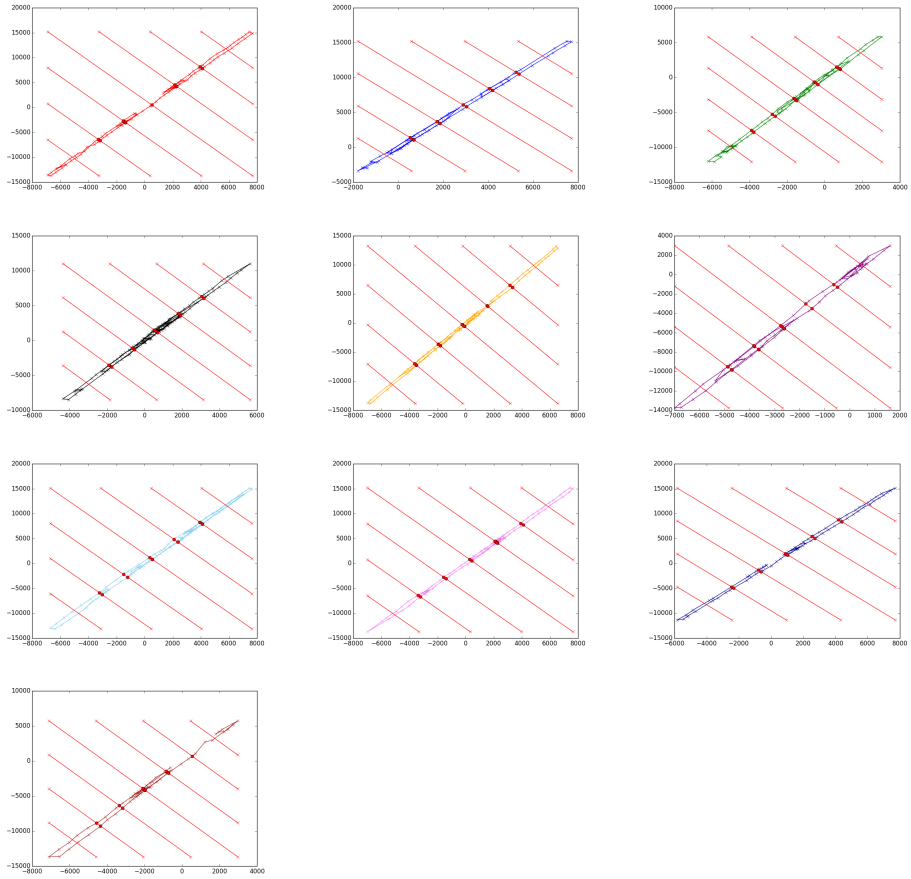
Obrázek 5.13: Vizualizace příčného kroku ve fázovém prostoru

počet průniků	prům. fázová rychlost
48	1003.19868149
56	1081.83426099
53	1237.81481524
24	1836.0785027
46	1213.23795865
22	1822.74807617
28	1853.63356246
28	1853.63356246
12	1818.78757135
42	1732.73541145

Obrázek 5.14: Příznakové vektory příčného kroku

5.2.2.7 Vyhodnocení

Vzorky s pádem kovadliny vykazují nejvyšší průměrnou fázovou rychlost a počet průsečíků s proloženou křivkou je konstantní. Nicméně podélný krok vykazuje v některých vzorcích



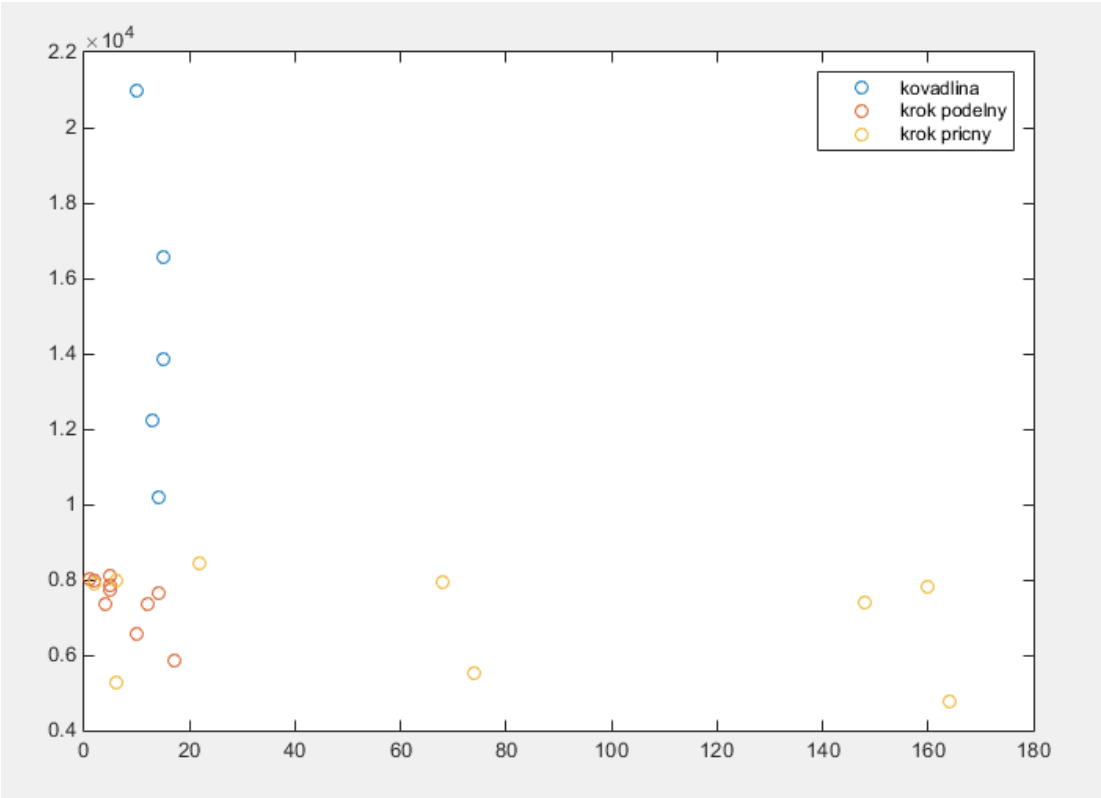
Obrázek 5.15: Vizualizace podélného kroku ve fázovém prostoru

počet průniků	prům. fázová rychlost
5	3637.03949457
11	2650.81277694
11	2703.79586927
14	1876.42443918
10	2879.99990178
10	2745.81557631
9	3528.59023369
6	3081.41668622
7	3375.15809461
9	2704.54652609

Obrázek 5.16: Příznakové vektory podélného kroku

stejnou průměrnou rychlost jako pád kovadliny, což je dáno rychlými záchvěvy v signálu. Na obrázku č.5.17 jsou vizualizovány příznakové vektory, kde jsou barevně odlišené jednotlivé vzorky. Můžeme vidět, že pád kovadliny se výrazně liší od detekovaného kroku. Ve fázi tvoření

příznakového vektoru jsme použili hodnotu maximální fázové rychlosti. Pokud budeme měřit průměrnou rychlost, pak zobrazení příznakových vektorů v prostoru se bude trochu lišit. Viz. obrázek č. 5.18. Vidíme, že se separují jednotlivé třídy do shluků, které potom lze detekovat algoritmem k-nejbližších sousedů.



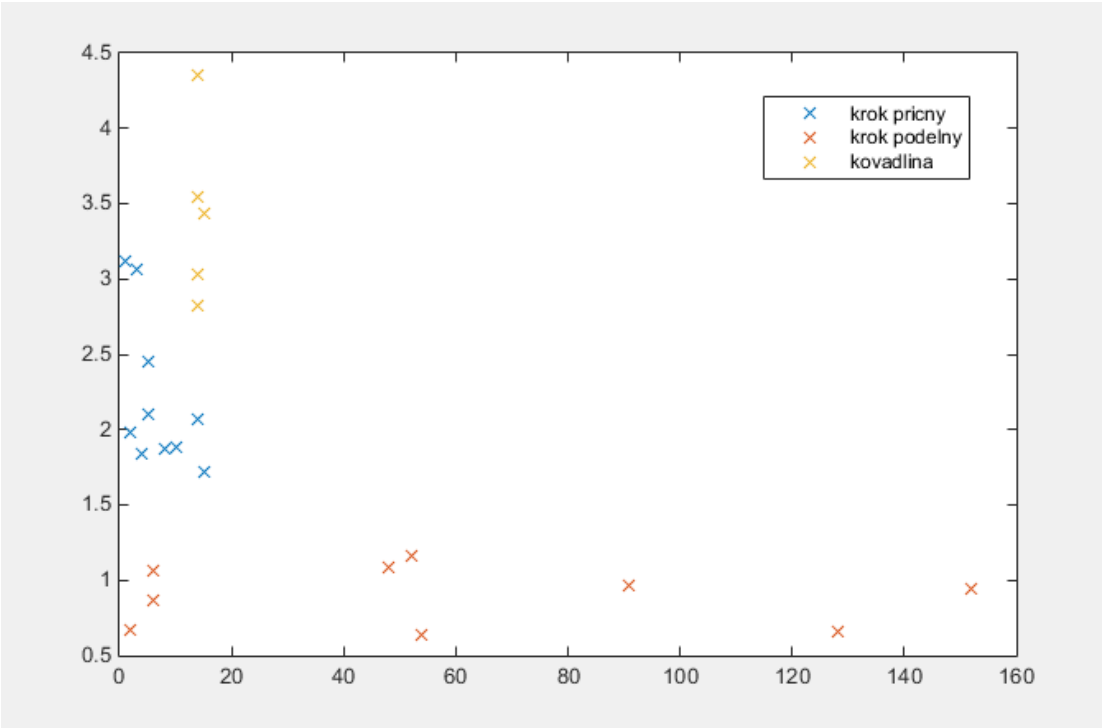
Obrázek 5.17: Vizualizace klasifikace jednotlivých vzorků v prostoru (max. fáz. rychlost)

5.2.3 Experiment #3

Experiment prováděný nad daty se zabývá zkoumáním detekovaných anomálií pomocí `DTW`. Teoretický postup je popsán v kapitole 3.7.2. Pro tvorbu příznakového vektoru použijeme průměrování matic a následně pro klasifikaci vektorů použijeme `SVM`, který je popsán v kapitole 3.7.5.

5.2.3.1 Popis

Analýza vzorků se provádí porovnáváním jednotlivých vzorků vůči sobě a následně se hledá podobnost mezi jednotlivými vzorky. Pro zkoumání podobnosti využijeme SVM, který by měl vzorek přiřadit do správné třídy. Tato metoda je oproti prvním dvěma experimentům časově náročná, protože každý nový vzorek je potřeba porovnat se všemi ostatními vzorky.



Obrázek 5.18: Vizualizace klasifikace jednotlivých vzorků v prostoru (prům. fáz. rychlost)

5.2.3.2 Vstupní data

Vstupními daty byly použité detekované anomálie ze signálů `Preprocessing`, konkrétně výstupy z filtrace vysokých a nízkých frekvencí. Pro experiment byly vybrány vzorky reprezentující jednotlivé události. Konkrétně následující vzorky:

- Krok - podélný (10 vzorků)
- Krok - příčný (10 vzorků)
- Pád kovadliny (5 vzorků)

5.2.3.3 Postup

Pro zpracování jednotlivých vzorků byla použita knihovna `numpy` a `mlpy`. Jednotlivé vzorky jsou porovnávány vůči sobě. Nejprve vypočteme vzdálenost mezi jednotlivými vzorky. pomocí následující funkce:

```
#use euclidian distances
for i in range(0,len(self.y1)):
    for j in range(0, len(self.y2)):
        distances[i,j] = (self.y2[j]-self.y1[i])**2
```

Potom si vypočteme akumulovanou energii mezi jednotlivými vzorky ze získané matice vzdáleností pomocí této funkce, kde nejprve inicializujeme první řádek a první sloupec a následně pak pomocí dynamického programování dopočteme zbývající hodnoty:

```
#accumulated cost
acc_cost = np.zeros((len(self.y1), len(self.y2))
acc_cost[0,0] = distances[0,0]
#axes x
for i in range(1, len(self.y1)):
    acc_cost[i,0] = distances[i,0] + acc_cost[i-1, 0]

#axes y
for i in range(1, len(self.y2)):
    acc_cost[0,i] = distances[0,i] + acc_cost[0, i-1]

for i in range(1, len(self.y1)):
    for j in range(1, len(self.y2)):
        acc_cost[i,j] = min(acc_cost[i-1, j-1], acc_cost[i-1, j],
        acc_cost[i, j-1]) + distances[i,j]
```

Jakmile máme matici akumulované energie, pak z této matice musíme vypočítat vektor. Protože SVM vyžaduje, aby délka vstupního vektoru měla vždy stejnou dimenzi, tak na vytvoření matice příznakového vektoru byla použita následující funkce:

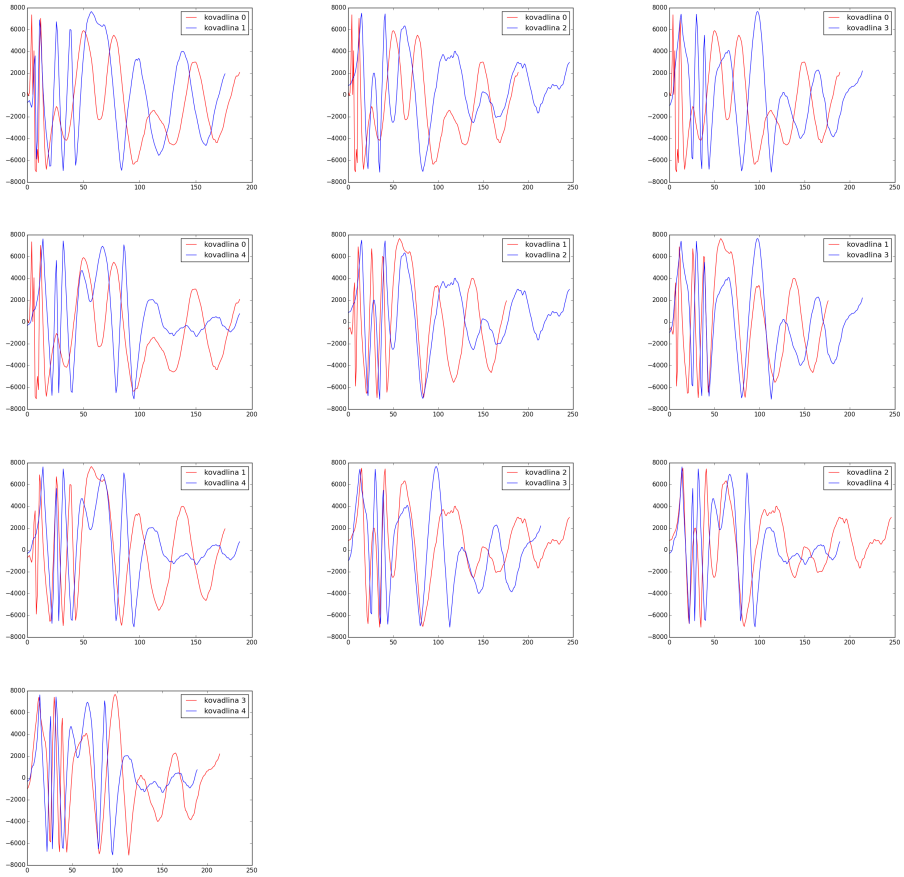
```
# dim1, dim2 - dimensions of vector matrix
for i in range(0, dim1):
    for j in range(0, dim2):
        avgval = self.submatrixmean(acc_cost[i*c:i*c+c,j*d:j*d+d])
        maxval = self.submatrixmax(acc_cost[i*c:i*c+c,j*d:j*d+d])
        vector_max[i,j] = maxval;
        vector_avg[i,j] = avgval;
...
    def submatrixmean(m):
        dim1 = len(m)
        dim2 = len(m[0])
        count = dim1*dim2
        value = 0;
        for i in range(0, dim1):
            for j in range(0, dim2):
                value += m[i,j]
        return value/count;

    def submatrixmax(m):
        dim1 = len(m)
        dim2 = len(m[0])
        value = 0;
        for i in range(0, dim1):
            for j in range(0, dim2):
                if m[i,j] > value:
                    value = m[i,j]
        return value;
```

Tyto vektory následně budou vstupem pro SVM. Protože SVM je algoritmus učení s učitelem, je nutné nejprve poskytnout data, podle kterých bude klasifikovat nové vzorky.

5.2.3.4 Kovadlina

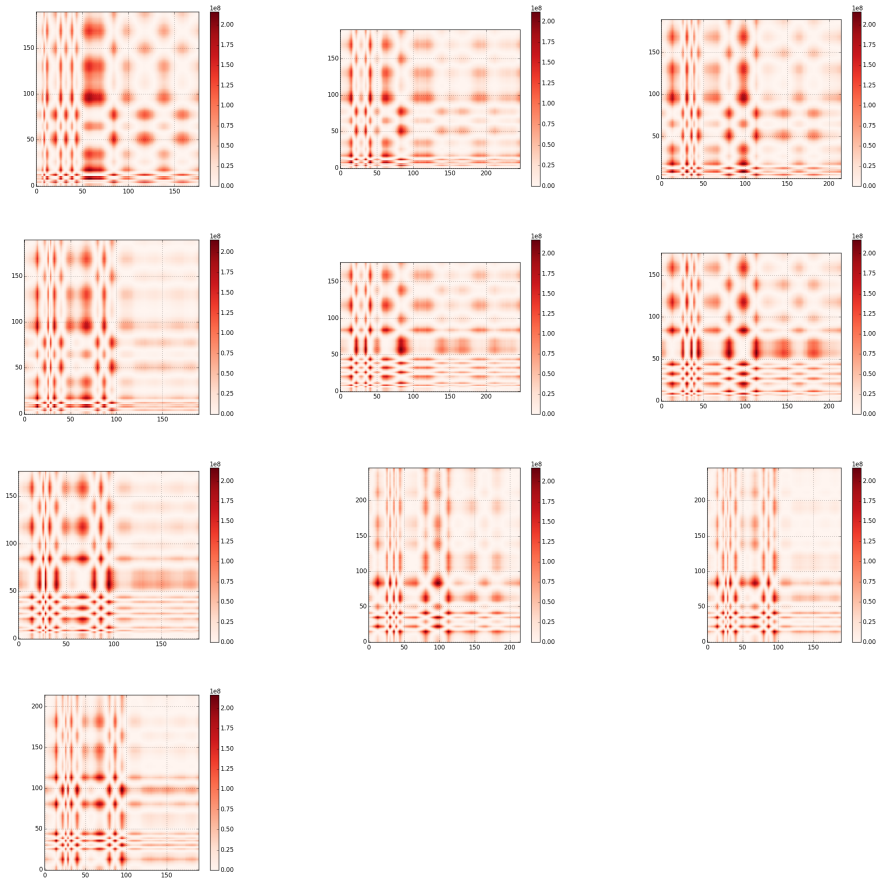
Dle postupu, který je popsán výše, byla provedena analýza na pěti různých vzorcích pádu kovadliny. Jejich reprezentace a porovnání je znázorněno na obrázku č. 5.19. Je vidět, že každý signál vykazuje jinou charakteristiku zákmitů a průběhu.



Obrázek 5.19: Porovnání naměřených vzorků pádu kovadliny.

Na dalším obrázku č. 5.20 jsou znázorněny jednotlivé vzdálenosti mezi body po jejich namapování. Již z naměřených vzdáleností je na obrázku patrné, že vzdálenosti mezi jednotlivými signály vykazují určitý pattern.

Dalším krokem bylo spočítání energie a cesty, které jsou znázorněny na obrázku č. 5.21. I zde je vidět určitý pattern v akumulované energii.



Obrázek 5.20: Vizualizace vzdáleností mezi jednotlivými vzorky.

5.2.3.5 Krok - příčný

Pro hledání patternu bylo použito deset nezávislých vzorků z experimentu. Jejich reprezentace a porovnání je znázorněno na obrázku č. B.1 (Pro ukázku pouze porovnání s prvním vzorkem). Zde je vidět určitá podobnost mezi jednotlivými signály.

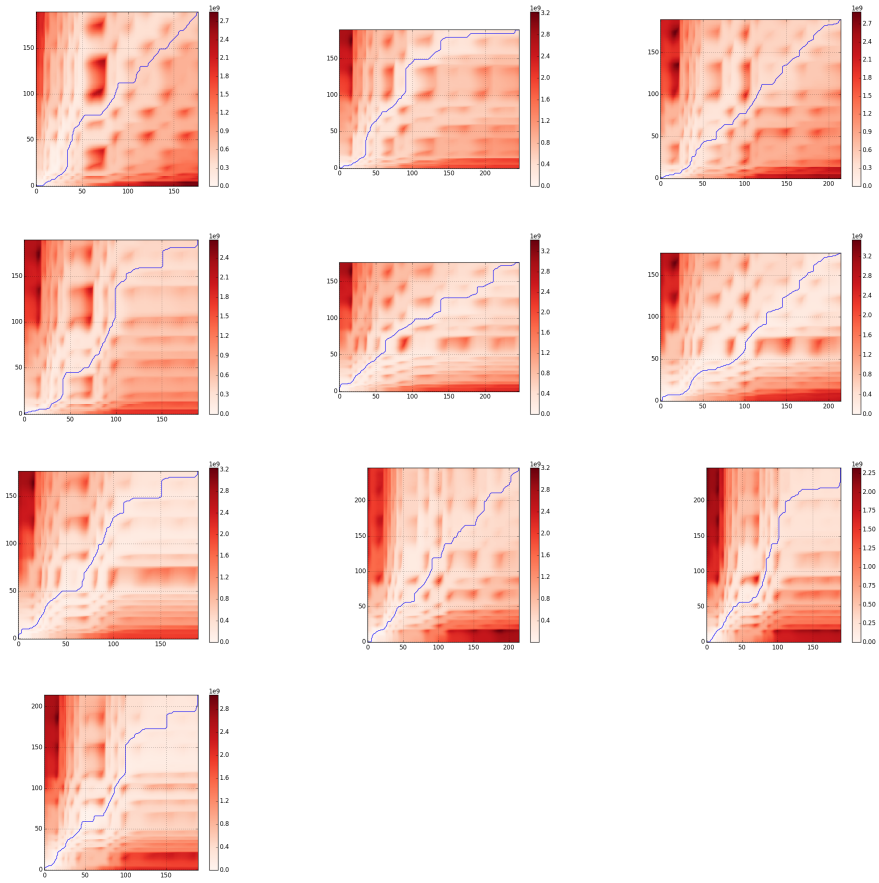
Na dalším obrázku č. B.2 jsou znázorněny jednotlivé vzdálenosti mezi body po jejich namapování. Na vizualizaci vzdáleností můžeme vidět určité podobnosti.

Dalším krokem bylo spočítání energie a cesty, které jsou znázorněny na obrázku č. B.3. Pattern akumulované energie vykazuje taktéž podobnost.

5.2.3.6 Krok - podélný

K analýze bylo použito deset nezávislých vzorků z experimentu. Jejich reprezentace a porovnání je znázorněno na obrázku č. B.4 (Pro ukázku pouze porovnání s prvním vzorkem). Zde je vidět určitá podobnost mezi jednotlivými signály.

Na dalším obrázku č. B.5 jsou znázorněny jednotlivé vzdálenosti mezi body po jejich namapování.



Obrázek 5.21: Vizualizace akumulované energie a nejlevnější cesty.

Dalším krokem bylo spočítání energie a cesty, které jsou znázorněny na obrázku č. B.6.

5.2.3.7 Validace

V předchozí části experimentu jsme se zabývali porovnáváním vzorků, které reprezentují stejnou anomálii. Tudíž je nutné prověřit, pokud porovnáme dva různé vzorky reprezentující různé anomálie, jestli budou vykazovat jiný pattern a budou se lišit, aby je bylo možné od sebe rozeznat. Pokud se lišit nebudou, tak metoda využívající metodu `DWT` nebude vhodná pro rozpoznávání anomálií.

Byla provedena validace všech různých anomálií vůči sobě (tzv. cross validace):

- Kovadlina vs. Krok příčný - výsledky jsou popsány v kapitole B.1.
- Kovadlina vs. Krok podélný - výsledky jsou popsány v kapitole B.2.
- Krok podélný vs. Krok příčný - výsledky jsou popsány v kapitole B.3.

5.2.3.8 SVM

Z výstupu vygenerujeme vstupní vektory pro SVM. Vstupní vzorky byly náhodně rozděleny na trénovací a testovací množinu. SVM se naučí na trénovací množině vstupní vektory a následně poté bude predikovat, kam zařadit nový vzorek, který ještě není oklasifikovaný.

Pro experiment byla použita implementace SVM z knihovny `sklearn`. S parametry `gamma=0.00000001` a `C=100000`. Pro klasifikaci byl použitý následný fragment kódu, který čte vstupní vektory ze souboru.

```
clf = svm.SVC(gamma=0.00000001, C=100000)
file = open('event-2[High pass]-mean-vectors-dim[10,10].json','r')
content = file.read()
data = json.loads(content)
#only even vectors are choosen to learning set
learningset = data[::2]
X = []
y = []
for d in learningset:
    X.append(d['vector'])
    y.append(d['class'])
#learning
clf.fit(X,y)
choosed = 3
#predicted class
print(clf.predict(data[choosed]['vector']))
#correct class
print(data[choosed]['class'])
```

5.2.3.9 Vyhodnocení

Vzorky zpracovávané metodou DWT byly převedeny do příznakových vektorů a následně tyto vektory byly klasifikovány pomocí SVM. Tato metoda byla testována na vektorech délky 36 a 100. Generování vektorů však je hodně časově náročné a tudíž by se nehodilo pro rozpoznávání v reálném čase. Výsledky automatického klasifikování pomocí SVM nedávaly dobré výsledky. Proces testování náhodně vybral skupinu vektorů, které použil jako trénovací data a pak byly vyhodnocovány vzorky, které SVM neznal. Výsledky přesnosti klasifikace se pohybovaly přibližně okolo 60% úspěšností. Úspěšnost závisela na počtu vzorků pádu kovadlin v trénovací množině. Pokud jich bylo více, tak SVM následně všechny vzorky z testovací množiny zařazoval do třídy pádu kovadliny.

5.2.4 Experiment #4

Poslední experiment se zabývá zkoumáním detekovaných anomálií za pomoci výpočtu kovariančních matic a následného biclusteringu, kterým budeme zkoumat podobnost mezi jednotlivými dvojicemi anomálií. Teoreticky je tato metoda popsána v kapitolách 3.7.6 a 3.7.7.

5.2.4.1 Popis

Samotná analýza se provádí porovnáváním jednotlivých vzorků vůči klidovému vzorku (vzorek, kdy není detekována žádná událost) a hledáme podobnost mezi vzorky za pomoci koeficientů kovarianční matice. Abychom byli schopní vzorky vůči sobě porovnávat je nutné vynechat anomálie, které jsou velmi krátké, protože by zkreslovaly výsledek. Pro zkoumání podobnosti použijeme metodu `biclusteringu`, která nám seskupí podobné vektory kovarianční matice, čímž zjistíme jednotlivé podobnosti nezi detekovanými anomáliemi.

5.2.4.2 Vstupní data

Vstupní data jsou anomálie detekované v `WVD`. Pro analýzu byly vybrány vysokofrekvenční vzorky. Pro experiment bylo použito 60 detekovaných anomálií.

5.2.4.3 Postup

K zpracování jednotlivých vzorků byly použity knihovny `numpy` a `sklearn`. Při načítání jednotlivých vzorků se odfiltrovávají krátké vzorky, které by zkreslovaly konečný výsledek, protože při porovnávání kovariančních matic je nutné, aby tyto matice měly stejnou dimenzi. Nejprve vybereme vhodné vzorky a vypočítáme kovarianční matice a uložíme jejch koeficienty použitím následující funkce:

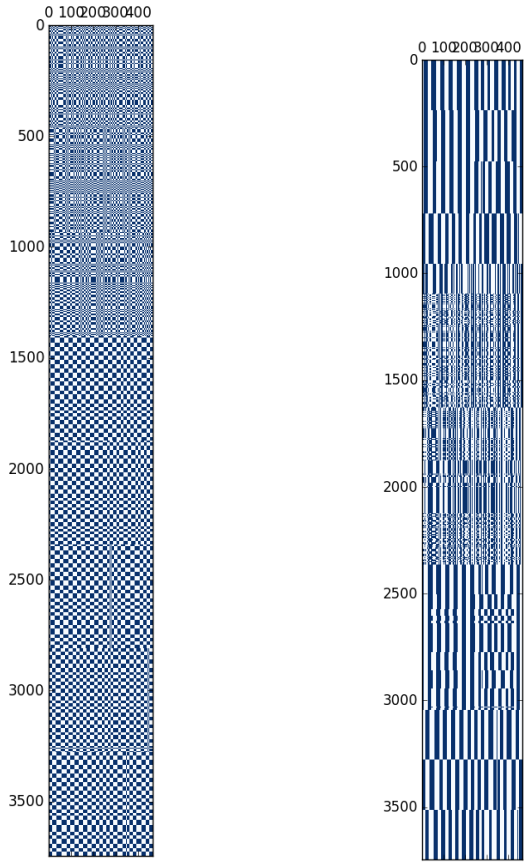
```
covariances = []
count = len(samples_y)
_min = sys.maxint
for i in range(0, count - 1):
    if _min > len(samples_y[i]):
       _min = len(samples_y[i])
           #adjust noise sample
       noise = self.noisesample[:_min]
       for i in range(0, count - 1):
            y = samples_y[i][:_min]
            z = zip(y, noise)
            cov = np.corrcoef(z)
            covariances.append(cov)
```

Nyní máme uložené koeficienty kovariančních matic. Poté je spojíme do velké matice a nad touto maticí spustíme spektrální biklastrovou analýzu viz. následující kód:

```
final = np.concatenate(covariances)
#spectral biclustering
model = SpectralBiclustering(n_clusters=(8,8), method='log', random_state=0)
model.fit(final)
fit_data = final[np.argsort(model.row_labels_)]
#result is in fit_data
```

5.2.4.4 Vyhodnocení

Experiment byl proveden s celkem 30 anomáliemi, které měly různou délku. Na základě vstupních podmínek byly vybrány jen ty vzorky, které obsahovaly alespoň 400 hodnot. Během experimentu bylo dosaženo nejlepších výsledkůpři použití vzorků z tabulky vysokofrekvenčního průchodu. Na obrázku č. 5.22 je vizualizována vstupní matice (vlevo) a výstupní matice po aplikaci biclusteringu (vpravo). Pro vyhodnocení clusterizace byla zkoumána permutace řádků ze vstupní matice, kde jednotlivé události jsou seřazeny za sebou. Při provádění experimentu tato metoda seskupovala anomálie, které vykazovaly podobné charakteristiky vůči klidovému signálu.



Obrázek 5.22: Vizualizace klasifikace pomocí biclusteringu

5.3 Diskuze výsledků

Jednotlivé experimenty byly realizovany nad dostupnými daty, která dodala společnost Safibra s.r.o. z experimentálního měření jednoho ze svých projektů Fibersense. Poskytnutá data obsahovala hodně šumu a vykazovala nestabilitu při opakovaných anomáliích stejného charakteru byly naměřené signály někdy hodně odlišné. Tato nestabilita komplikovala celý proces rozpoznávání, takže bylo zapotřebí hledat takové metody, které jsou schopné najít podobnost mezi takovými anomáliemi.

V rámci diplomové práce byly navrženy a zrealizovány celkem čtyři experimenty. Jednotlivé metody zpracování byly navrhovány s ohledem na charakter vstupních dat. Nejprve bylo nutné najít charakteristické vlastnosti, podle kterých by bylo možné rozlišit a propojit mezi sebou podobné detekované anomálie. Během navrhování jednotlivých experimentů bylo zapotřebí používat již vzbrané známé postupy a upravovat jim parametry tak, aby je bylo možné aplikovat na tuto diplomovou práci.

V předchozích kapitolách byly představeny jednotlivé navržené experimenty, jejich postup a následně i vyhodnocení.

- **Experiment 1** - Tento experiment vycházel z myšlenky, že příznakový vektor můžeme čerpat z proložené křivky signálem. Nicméně po získání koeficientů proložených křivek tato data nevykazovala žádnou podobnost. Proto bych tuto metodu nedoporučoval dále rozvíjet.

- **Experiment 2** - V tomto experimentu jsou anomálie zkoumané ve fázovém prostoru. V každé anomálii se zkoumá rychlost změn signálu, která generuje jeden atribut příznakového vektoru. Druhým atributem je detekce, kolikrát signál přešel přes uměle vytvořené prahy. V tomto experimentu jednotlivé anomálie vykazovaly na první pohled dobře odlišitelné křivky. Tuto metodu je možné dále do budoucnosti rozvíjet a vylepšovat.

- **Experiment 3** - Zde je použitá metoda pro detekci podobnosti zvukového signálu DTW. Tato metoda umí eliminovat různé rychlosti a fázový posun signálu. Nevýhodou je velký příznakový vektor, který byl potřeba pro rozlišování jemných rozdílů mezi signály. Pro klasifikaci byl použit SVM, který vykazoval přesnost klasifikace na 50%. Takže tato metoda není vhodná pro budoucí práci s anomáliemi.

- **Experiment 4** - Poslední experiment se zabývá poměřováním jednotlivých anomálií vůči klidovému stavu signálu. Používá k tomu výpočtu kovariančích matic. Tyto matice jsou poté shlukovány podle podobnosti koeficientů. Tato metoda je také vhodná pro další práci s anomáliemi.

Kapitola 6

Závěr

Cílem diplomové práce byla implementace aplikace pro grafickou vizualizaci naměřených dat z experimentů prováděných společností Safibra s.r.o.. Dálším bodem byla detekce anomálií v naměřeném signálu a následná klasifikace nebo shlukování detekovaných anomálií tak, aby podobné anomálie byly zařazeny do stejné třídy či shluku pomocí experimentů.

V kapitole 2 byla popsána struktura opticko vláknové senzorové sítě, jak je technicky řešená a jak probíhá snímání signálu. Dále je zde popsán postup předzpracování získaného signálu pro aplikaci grafické vizualizace a následný datamiming. Závěr kapitoly představuje princip detekce anomálií v předzpracovaných datech pomocí prahového detektoru.

Kapitola 3 se zaměřuje na popis řešeného problému. Zde je podrobněji popsáno řešení problému včetně popisu vstupních dat. Hlavní částí této kapitoly je teoretický popis použitých metod pro rozpoznávání jednotlivých anomálií.

Další kapitola 4 je věnována popisu aplikace grafické vizualizace, jaké vznikaly požadavky na aplikaci a jak probíhala samotná implementace aplikace. Součástí je popis analýzy, řešení stěžejních částí aplikace, uživatelský manuál a její samotné testování. Tato kapitola může být použita pro budoucí reimplementaci aplikace.

Poslední kapitola 5 popisuje samotný průběh provedených experimentů nad detekovanými anomáliemi. V každém experimentu je podrobný popis, jsou zde uvedeny fragmenty kódu včetně použitých knihoven, vstupní parametry a vyhodnocení výsledků. Na závěr kapitoly se nachází diskuze výsledků, která shrnuje průběh všech experimentů a shrnuje jejich výhody a nevýhody.

6.1 Shrnutí

V rámci diplomové práce byly analyzovány požadavky od společnosti Safibra s.r.o. na vývoj aplikace pro grafickou vizualizaci naměřených dat z interně prováděných experimentů. Byl vyvinut protoyp aplikace v jazyce Python, který na základě konfiguračního souboru zobrazí uživateli požadovaná data.

Součástí práce bylo otestování navrhnutých dataminingových metod nad množinou detekovaných anomálií. V rámci práce byly navrženy celkem čtyři postupy, které zahrnují vytvoření příznakového vektoru, následné shlukování/klasifikace dat a jejich reprezentace. Uživatel

má možnost jednotlivé anomálie manuálně oklasifikovat, čímž pak může kontrolovat, jestli rozpoznávací algoritmus funguje správně.

Po provedení experimentu bylo zjištěno, že pouze dva ze čtyř navržených experimentů jsou schopny pracovat s detekovanými anomáliemi a klasifikovat/shlukovat je s přijatelnou pravděpodobností. Konkrétně experiment zabývající se analýzou ve fázovém prostoru a kovariančními maticemi. Tyto metody je možné použít k dalšímu vývoji a dále je vylepšit.

Literatura

[1] *Data Mining, What is Data Mining?* [online]. [cit. 18. 3. 2016]. Dostupné z: <http://www.anderson.ucla.edu/faculty/jason.frand/teacher/technologies/palace/datamining.htm>.

[2] *Fiber optics* [online]. [cit. 25. 3. 2016]. Dostupné z: <http://www.slideshare.net/lpapadop/fiber-optics-23738146>.

[3] *Fiber SenSys, Inc.* [online]. [cit. 18. 3. 2016]. Dostupné z: <http://www.fibersensys.com/>.

[4] *Safibra s.r.o* [online]. [cit. 18. 3. 2016]. Dostupné z: <http://www.safibra.cz>.

[5] *Algoritmy podpůrných vektorů* [online]. [cit. 05. 5. 2016]. Dostupné z: <https://is.muni.cz/el/1433/podzim2006/PA034/09_SVM.pdf>.

[6] Abarbanel H. D. I. *Analysis of observed chaotic data.* Springer, New York, 1996.

[7] Hansen, B.E. *Nearest Neighbour Methods.* University of Winsconsin-Madison, 2009.

[8] Hongya Zhaoa, Alan Wee-Chung Liewb, Doris Z. Wangc, and Hong Yan. Biclustering Analysis for Pattern Discovery: Current Techniques, Comparative Studies and Applications. *BENTHAM SCIENCE PUBL LTD.* 2012.

[9] Horák J. and Krlín L. and Raidl A. *Deterministický chaos a jeho fyzikální aplikace.* Academia, 2003.

[10] Josef Tvrdík. *Analýza vícerozměrných dat.* Ostrava, 2003.

[11] Šíma, Jiří, Neruda, Roman. *Teoretické otázky neuronových sítí.* MATFYZPRESS, 1996.

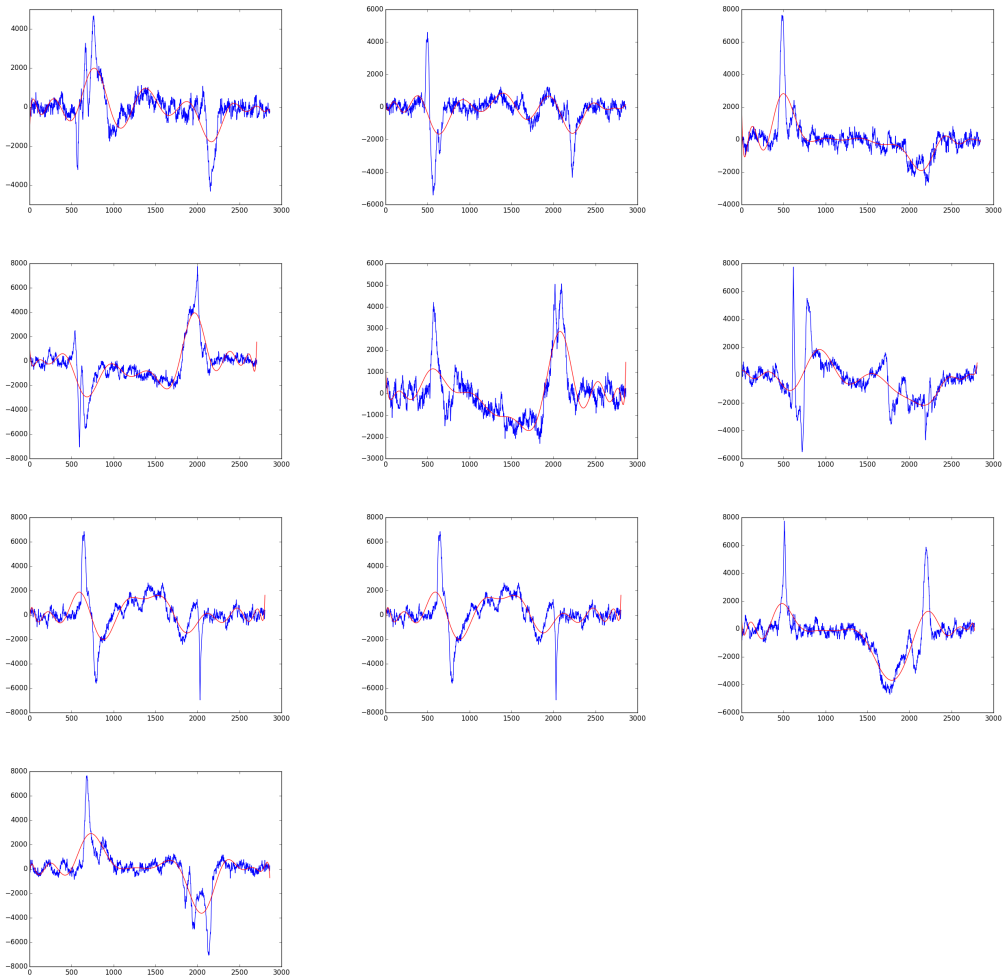
[12] Packard N., Crutchfield J., Farmer D., Shaw R. *Geometry from a time series.* Phys. Rev. Lett, 1980. p. 712.

[13] Stan Z. Li, Anil Jain. *Dynamic Time Warping (DTW).* Springer US, 2009.

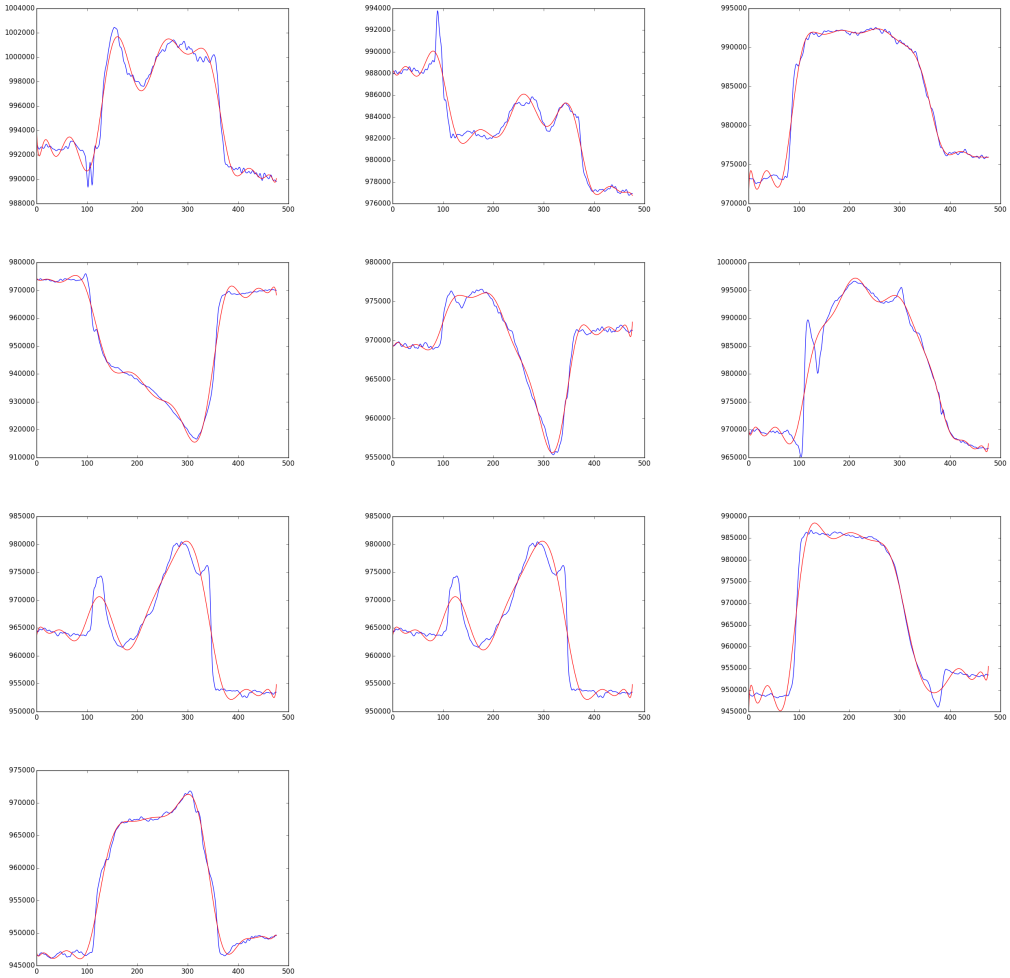
[14] Stan Z. Li, Anil Jain. *Fiber Optic Sensors, Second Edition.* CRC Press, 2008.

Příloha A

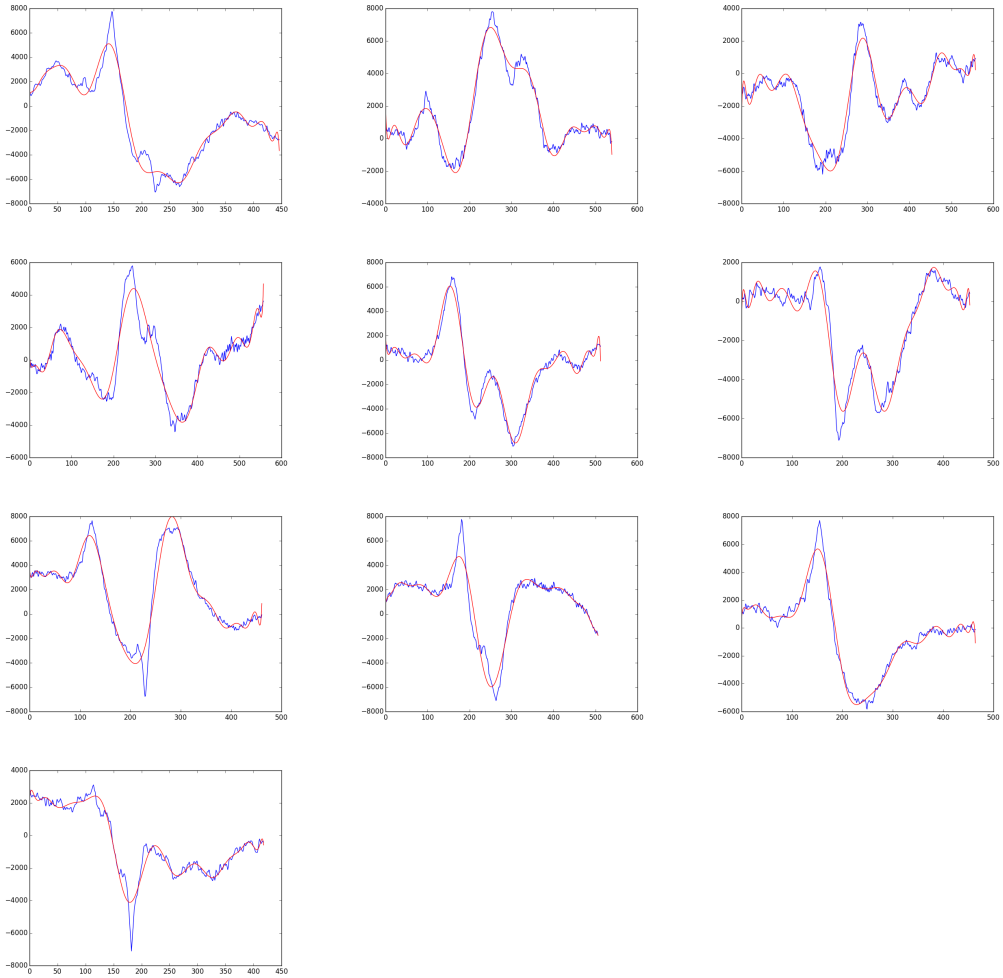
Experiment #1



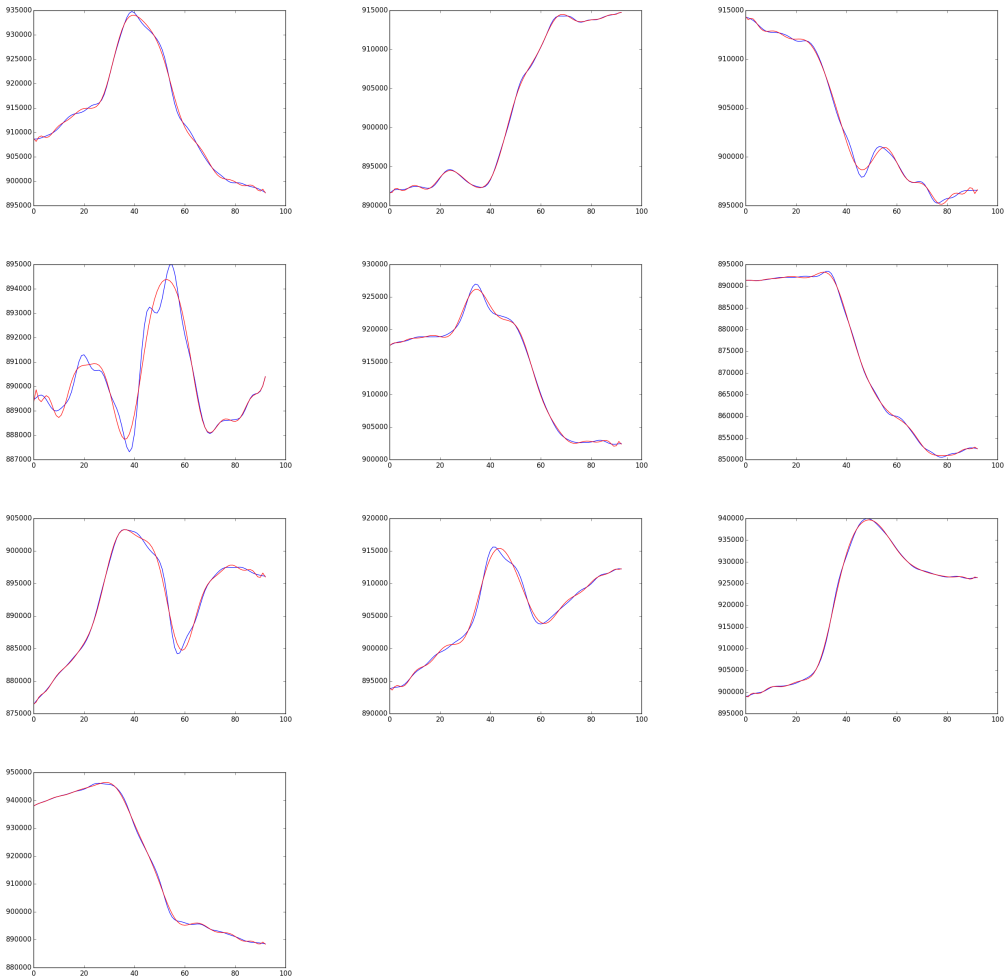
Obrázek A.1: Jednotlivé vzorky vysokofrekvenčního průchodu pro příčný krok



Obrázek A.2: Jednotlivé vzorky z nízkofrekvenčního průchodu pro příčný krok



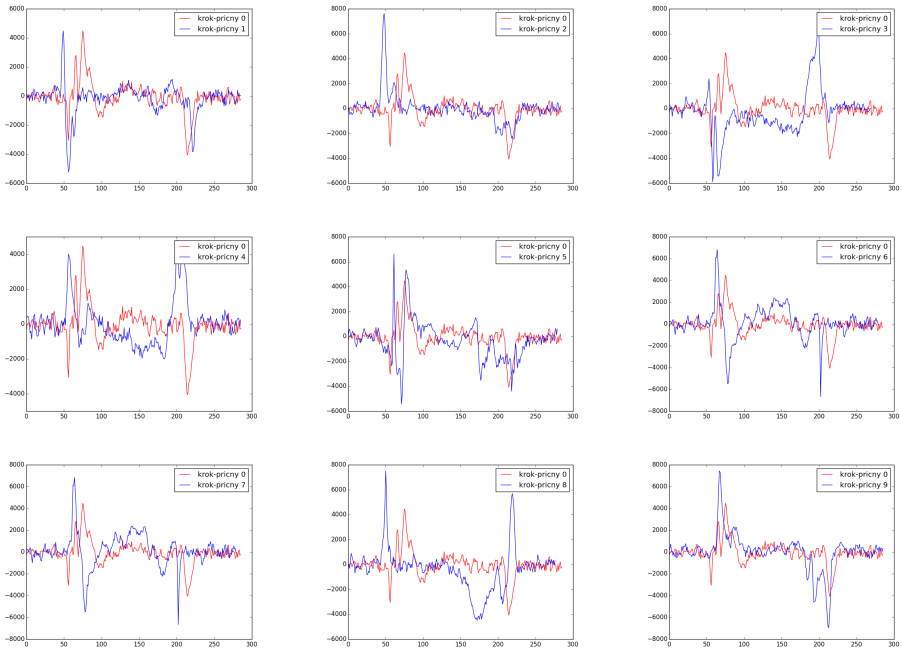
Obrázek A.3: Jednotlivé vysokofrekvenční vzorky pro podélný krok



Obrázek A.4: Jednotlivé nízkofrekvenční vzorky pro podélný krok

Příloha B

Experiment #3

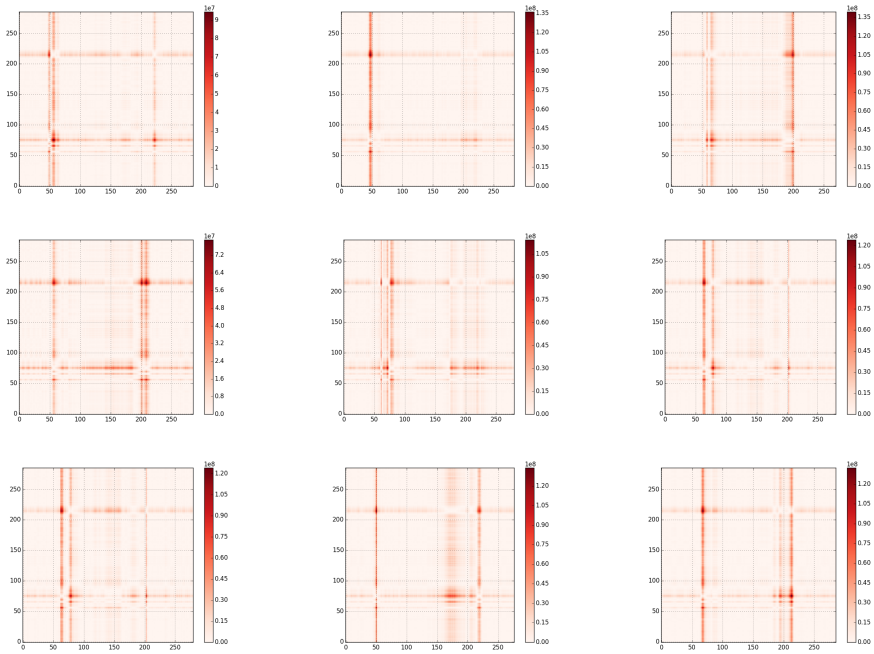


Obrázek B.1: Porovnání naměřených vzorků příčných kroků

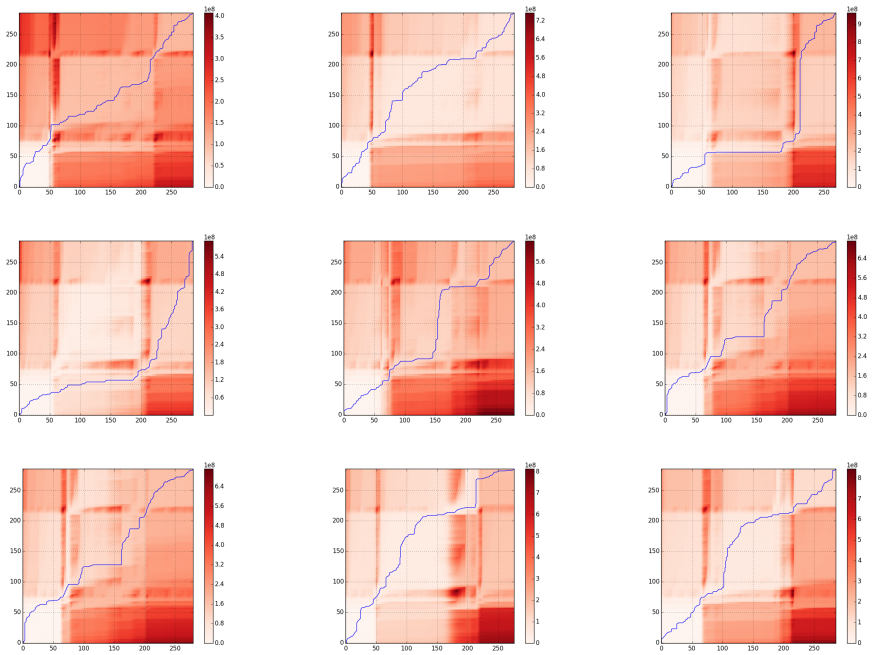
B.1 Kovadlina vs. Krok příčný

Provedeme stejný postup porovnávání vzorků jako při porovnávání jednotlivých vzorků, když byly porovnávány vůči sobě.

Na obrázku č. B.7 je vizualizace jednotlivých vzorků, na kterých budeme provádět validaci. Na dalším obrázku č. B.8 je vizualizace vzdáleností po namapování signálu. Na obrázku č. B.9 je vizualizace energie akumulované porovnávanými vzorky.

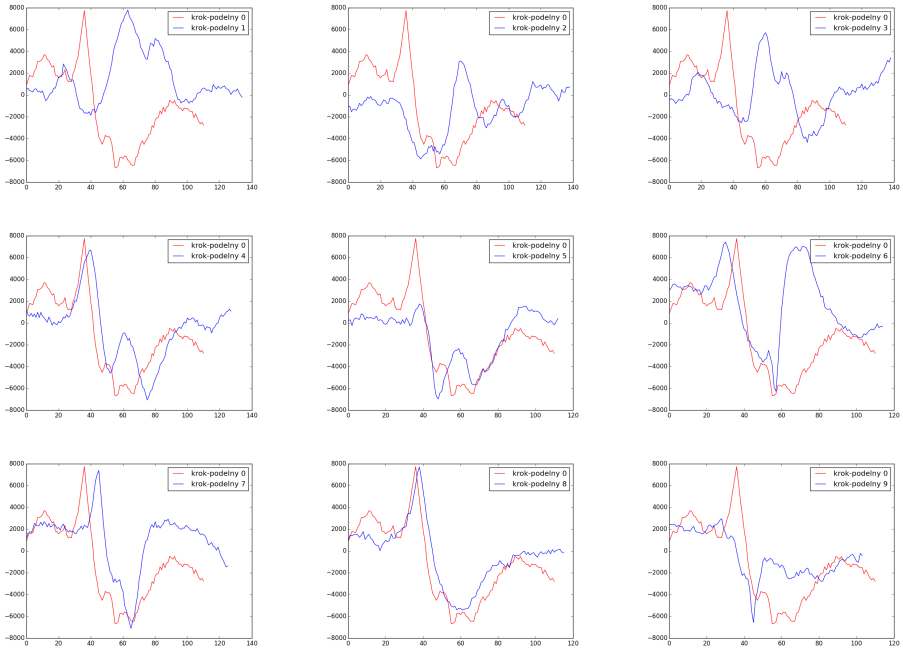


Obrázek B.2: Vizualizace vzdáleností mezi jednotlivými vzorky příčného kroku

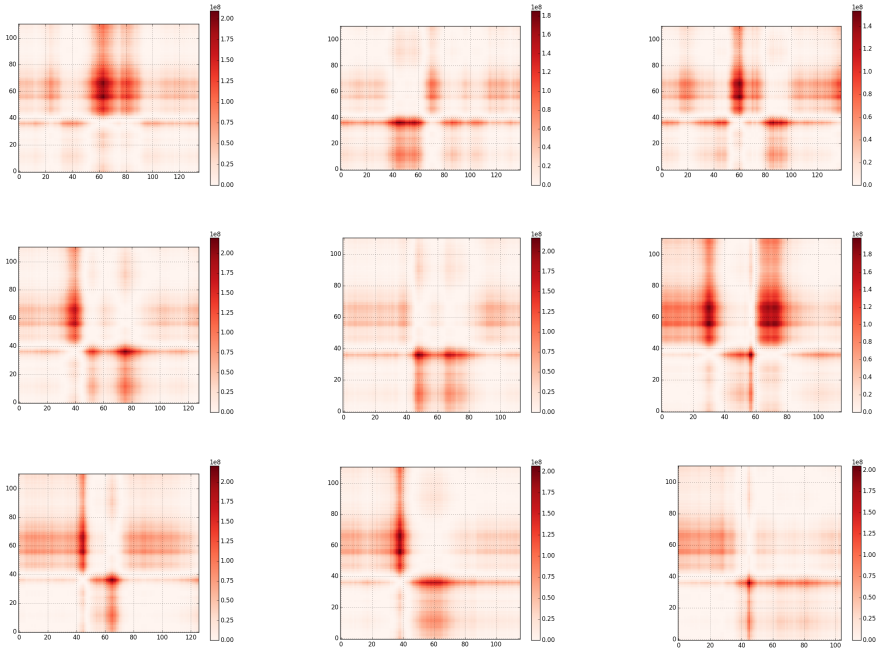


Obrázek B.3: Vizualizace akumulované energie a nejlevnější cesty příčného kroku

Při porovnání obrázků č. 5.21 a obrázku č. B.9 je možné vidět rozdíl v heatmapě. Ve výsledku vznikla odlišná vizualizace otisku akumulované energie. Je tedy možné od sebe

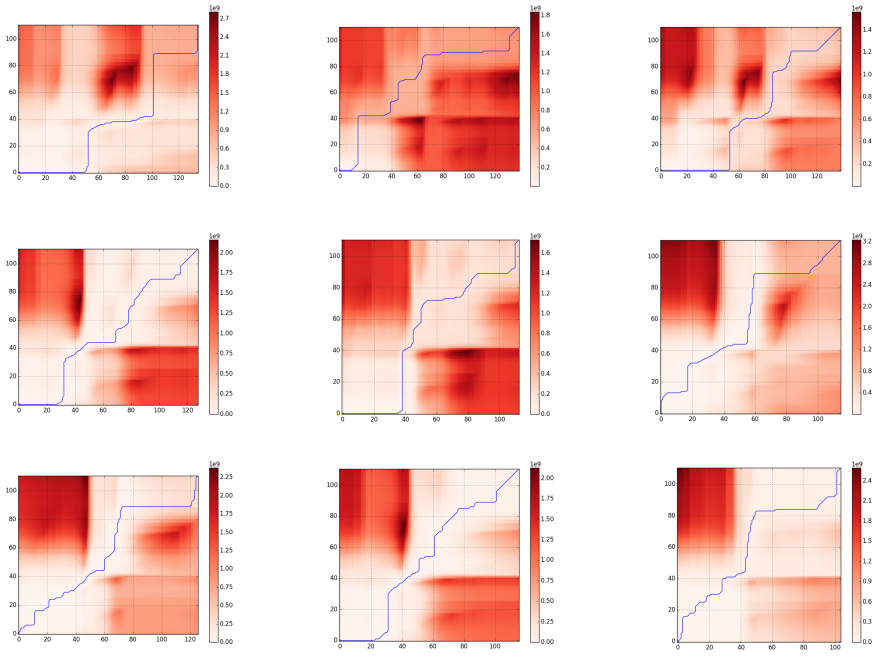


Obrázek B.4: Porovnání naměřených vzorků podélných kroků



Obrázek B.5: Vizualizace vzdáleností mezi jednotlivými vzorky podélných kroků

odlišit pád kovadliny a příčný krok.



Obrázek B.6: Vizualizace akumulované energie a nejlevnější cesty podélných kroků

B.2 Kovadlina vs. Krok podélný

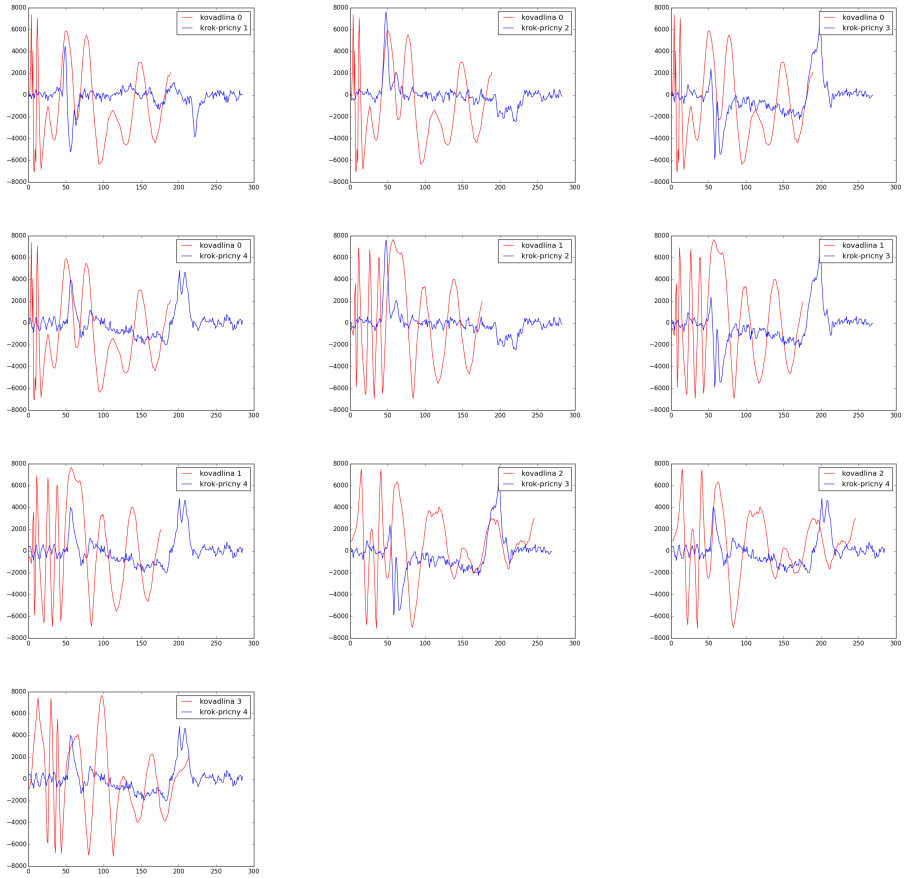
Na obrázku č. B.10 je vizualizace jednotlivých vzorků, na kterých budeme provádět validaci. Na dalším obrázku č. B.11 je vizualizace vzdáleností po namapování signálu. Na obrázku č. B.12 je vizualizace energie akumulované porovnávanými vzorky.

Při porovnání obrázků č. 5.21 a obrázku č. B.12 vidíme patrný rozdíl v heatmapě. Ve výsledku vznikla také odlišná vizualizace otisku akumulované energie. Je tedy možné od sebe odlišit pád kovadliny a podélný krok.

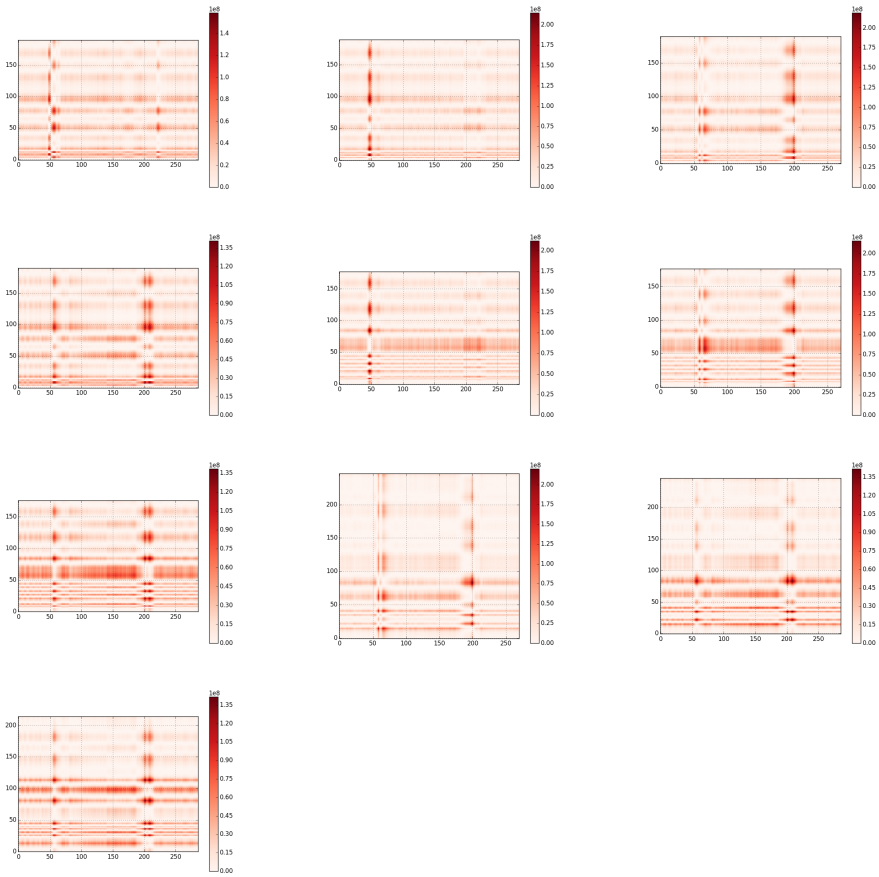
B.3 Krok podélný vs. Krok příčný

Na obrázku č. B.13 je vizualizace jednotlivých vzorků, na kterých budeme provádět validaci. Na dalším obrázku č. B.14 je vizualizace vzdáleností po namapování signálu. Na obrázku č. B.15 je vizualizace energie akumulované porovnávanými vzorky.

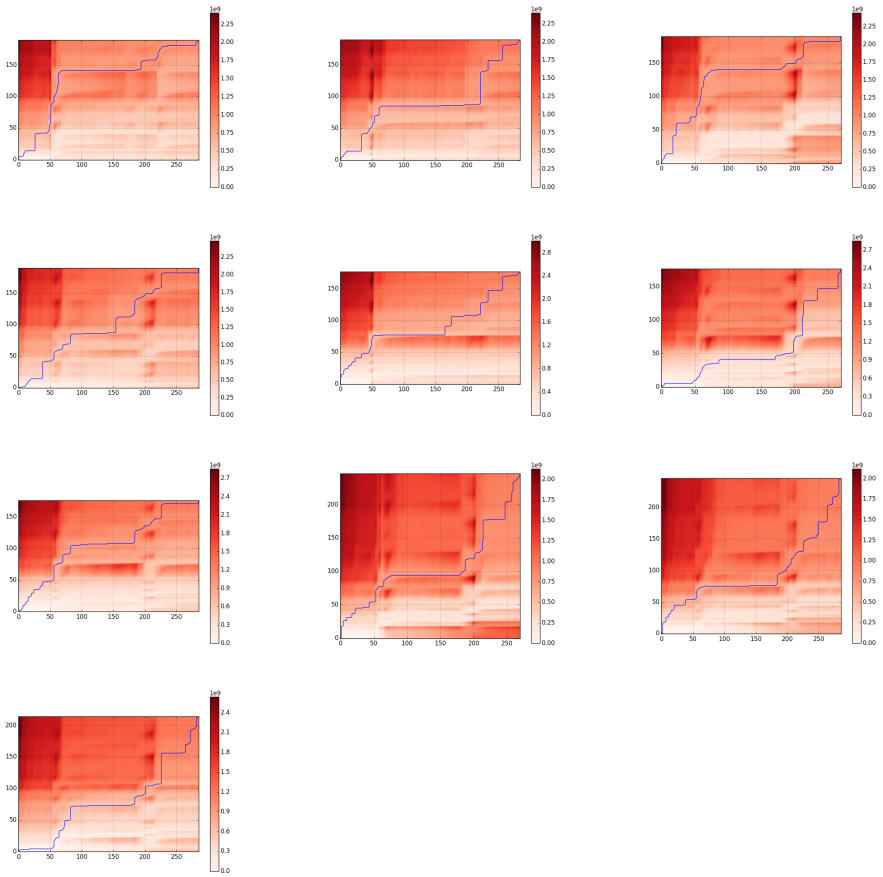
Při porovnání obrázků č. B.6 a obrázku č. B.15 vidíme rozdíl v heatmapě. Máme také odlišnou vizualizaci otisku akumulované energie, která má podobné rysy jako krok podélný. Nicméně je tedy možné od sebe odlišit pád, krok příčný a podélný.



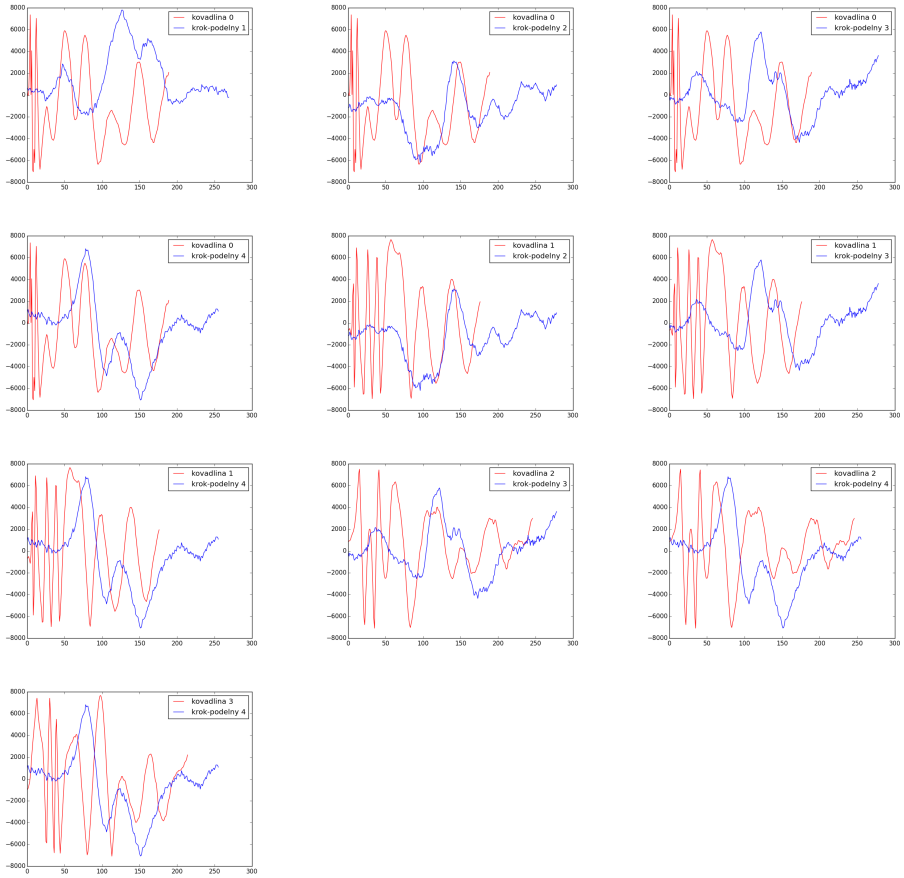
Obrázek B.7: Porovnání naměřených vzorků příčných kroků a pádů kovadliny



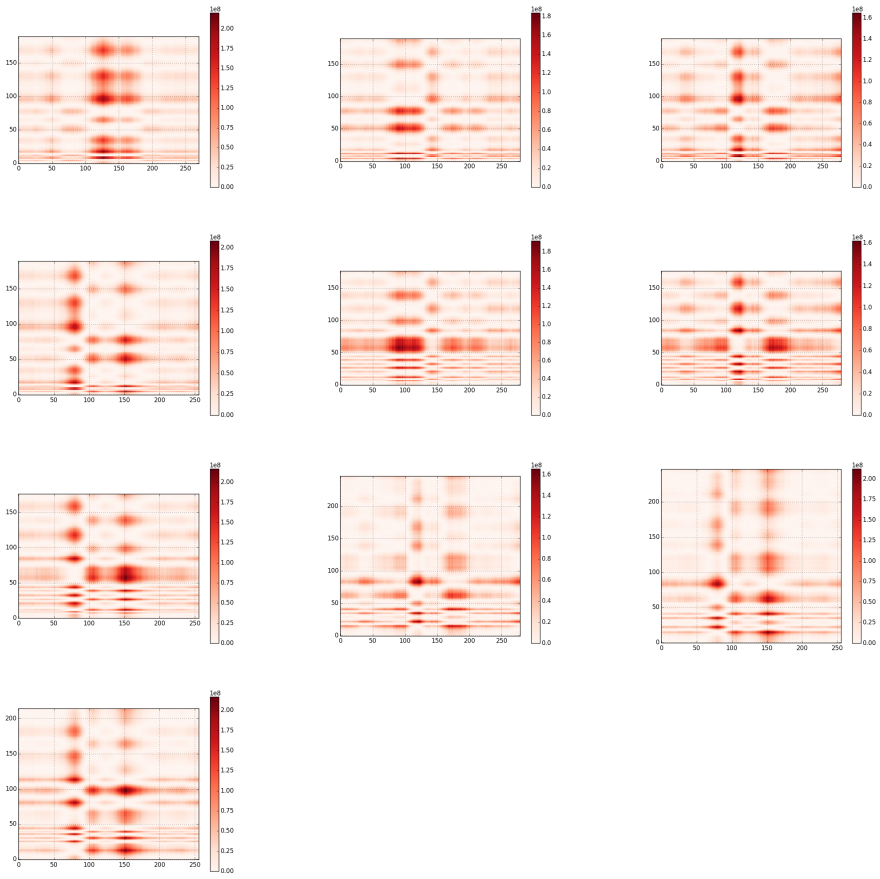
Obrázek B.8: Vizualizace vzdáleností mezi jednotlivými vzorky příčných kroků a pádů kovadliny



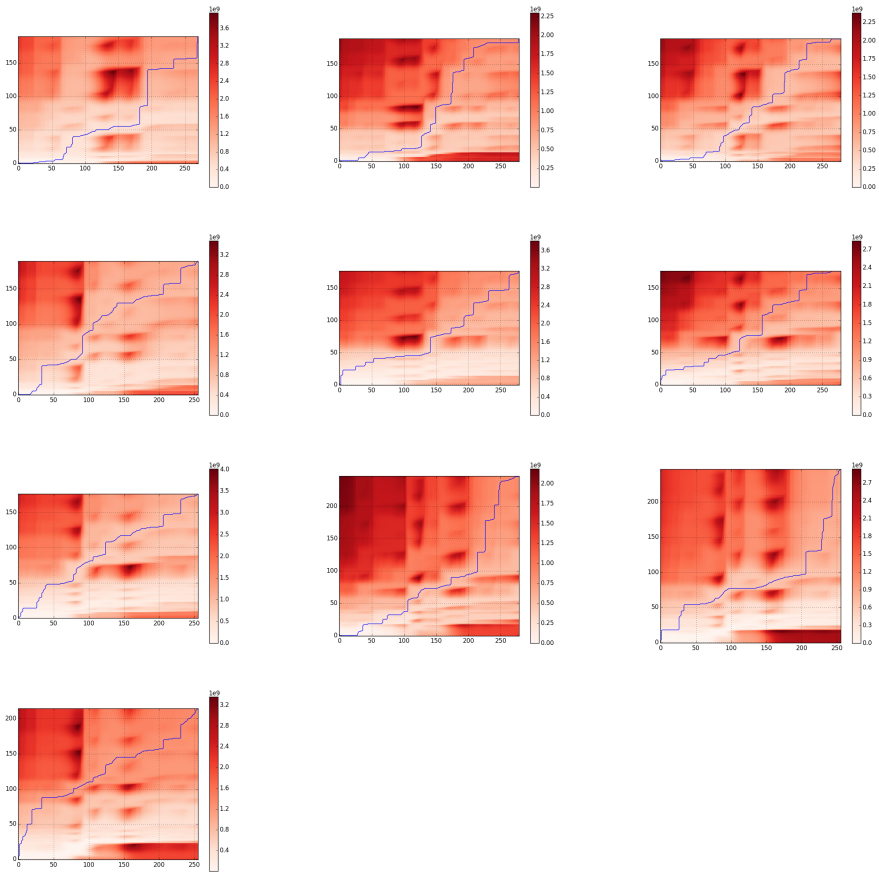
Obrázek B.9: Vizualizace akumulované energie a nejlevnější cesty příčných kroků a pádů kovadliny



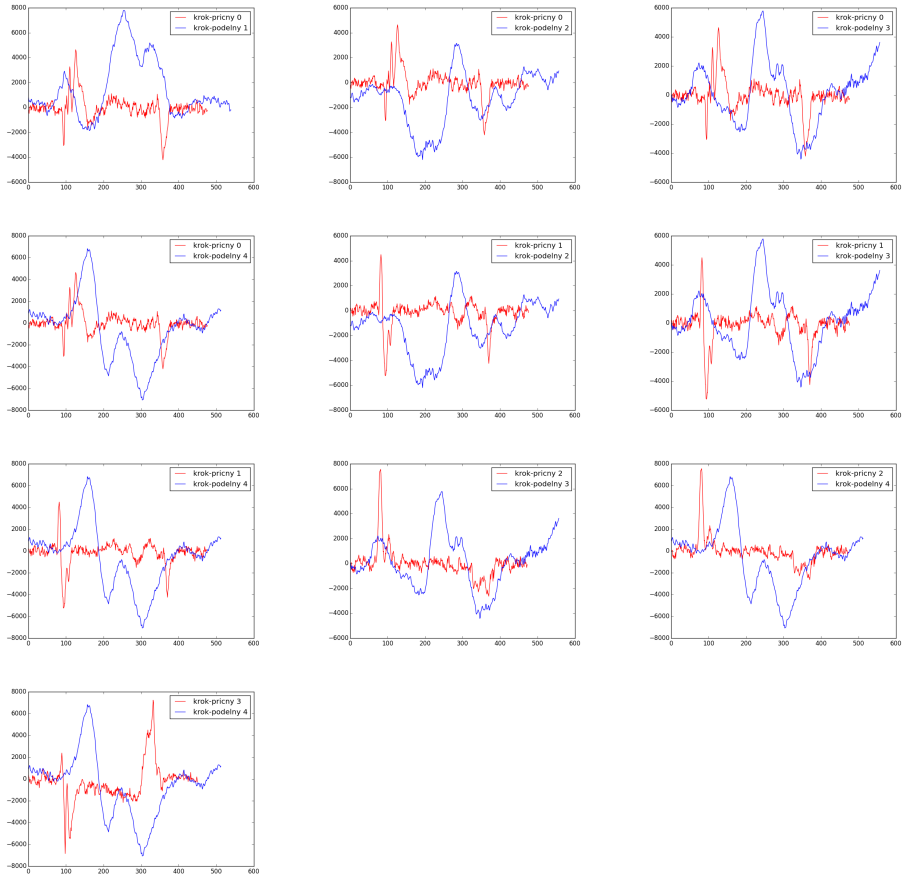
Obrázek B.10: Porovnání naměřených vzorků podélných kroků a pádů kovadliny



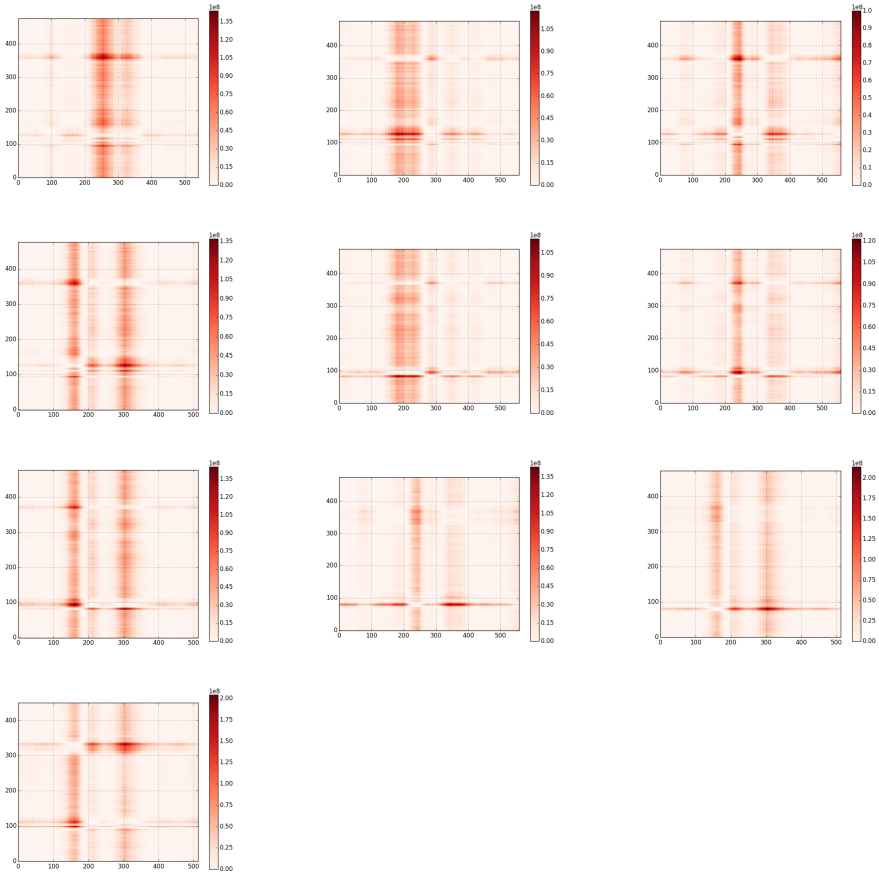
Obrázek B.11: Vizualizace vzdáleností mezi jednotlivými vzorky podélných kroků a pádů kovadliny



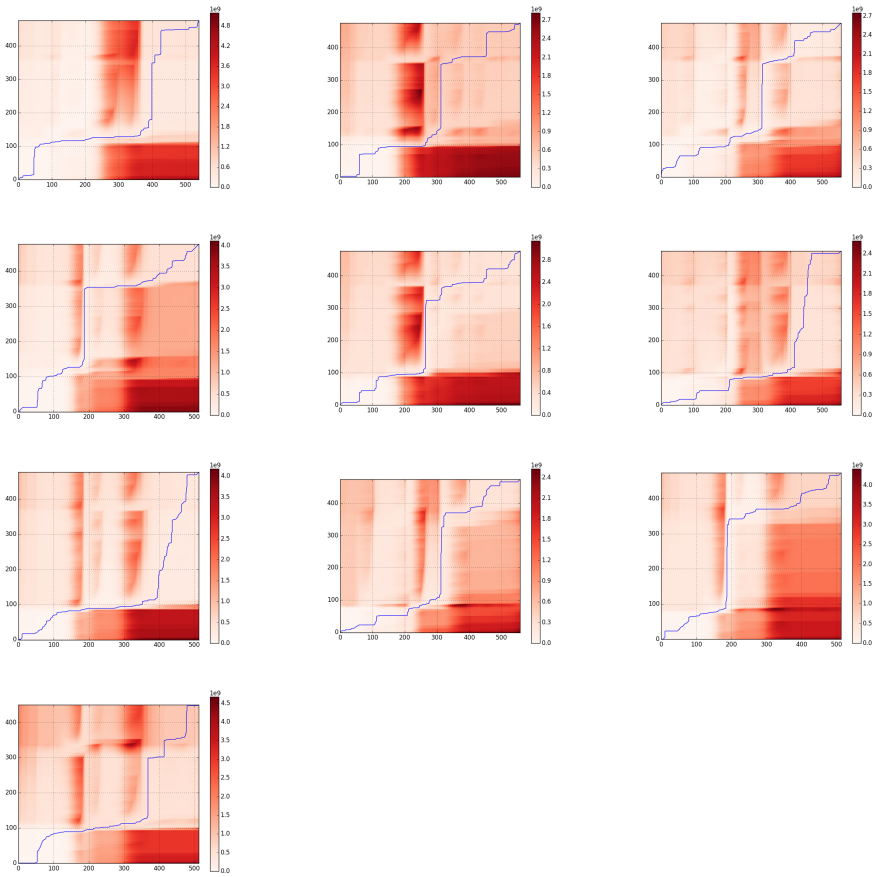
Obrázek B.12: Vizualizace akumulované energie a nejlevnější cesty podélných kroků a pádů kovadliny



Obrázek B.13: Porovnání naměřených vzorků podélných a příčných kroků



Obrázek B.14: Vizualizace vzdáleností mezi jednotlivými vzorky podélných a příčných kroků



Obrázek B.15: Vizualizace akumulované energie a nejlevnější cesty podélných a příčných kroků

Příloha C

Instalační příručka

V této části je popis, jak aplikaci uvést do provozu tak, aby ji bylo možné začít používat. Dále je zde popis konfiguračního XML, který slouží pro nastavení prostředí aplikace grafické vizualizace.

C.1 Konfigurace

Validační dokument konfiguračního XML souboru pro zobrazované widgety.

```
<!DOCTYPE guardsence [
<!ELEMENT guardsence (general, graphwidget, eventwidget, imagewidget,
videowidget)>
<!-- obecne nastaveni aplikace -->
<!ELEMENT general (h5filepath, startinterval, maxzoomhelperlimit,
marginzoomhelper, exportimagedpi, timefromzero)>
<!-- cesta k H5 souboru -->
<!ELEMENT h5filepath (#PCDATA)>
<!-- pocatecni interval signalu k zobrazeni -->
<!ELEMENT startinterval (#PCDATA)>
<!-- horni limit intervalu pro aplikaci automatickeho zoomu -->
<!ELEMENT maxzoomhelperlimit (#PCDATA)>
<!-- zobrazeni signalu pred a po urcite anomalie(hodnota v procentech) -->
<!ELEMENT marginzoomhelper (#PCDATA)>
<!-- DPI exportovanych screenshotu -->
<!ELEMENT exportimagedpi (#PCDATA)>
<!-- deleni casove osy (pri presazeni max. hod. int) -->
<!ELEMENT timefromzero (#PCDATA)>

<!ELEMENT graphwidget (metadata, graphplots)>
<!ELEMENT metadata (name, xplotlabel, yplotlabel, offset)>
<!-- nazev widgetu -->
<!ELEMENT name (#PCDATA)>
<!-- nazev X osy -->
<!ELEMENT xplotlabel (#PCDATA)>
<!-- nazev Y osy -->
<!ELEMENT yplotlabel (#PCDATA)>
```

```
<!-- posunuti signalu po Y ose -->
<!ELEMENT offset (#PCDATA)>
<!-- reprezentace signalu ve widgetu -->
<!ELEMENT graphplots (graphplot)>
<!ELEMENT graphplot (tablename, xcolname, ycolname,
percentile, divider, step, color)>
<!-- cesta k tabulce v H5 souboru -->
<!ELEMENT tablename (#PCDATA)>
<!-- nazev sloupce pro X hodnoty -->
<!ELEMENT xcolname (#PCDATA)>
<!-- nacteni X dat z jine tabulky -->
<!ATTLIST xcolname crosstable tablename CDATA>
<!-- nazev sloupce pro Y hodnoty -->
<!ELEMENT ycolname (#PCDATA)>
<!-- pocatecni hodnota percentil (v procentech) -->
<!ELEMENT percentil (#PCDATA)>
<!-- vydeleni signalu konkretni hodnotou -->
<!ELEMENT divider (#PCDATA)>
<!-- cteni dat z tabulky s urcitym skokem -->
<!ELEMENT step (#PCDATA)>
<!-- barva krivky -->
<!ELEMENT color (#PCDATA)>

<!ELEMENT eventwidget (tablename, name, columnsnames, highlight)>
<!-- id widgetu -->
<!ATTLIST eventwidget id #REQUIRED>
<!-- cesta k tabulce v H5 souboru -->
<!ELEMENT tablename (#PCDATA)>
<!-- nazev widgetu -->
<!ELEMENT name (#PCDATA)>
<!-- zvyrazneni intervalu anomalie ve widgetu -->
<!ELEMENT highlight (#PCDATA)>
<!-- barva intervalu -->
<!ATTLIST highlight color CDATA>
<!ELEMENT columnsnames (id, timestartoff, timeendoff,
timelength, indexrawstart, indexrawend)>
<!-- nazev sloupce pro ID -->
<!ELEMENT id (#PCDATA)>
<!-- nazev sloupce pro zacatek anomalie -->
<!ELEMENT timestartoff (#PCDATA)>
<!-- nazev sloupce pro konec anomalie -->
<!ELEMENT timeendoff (#PCDATA)>
<!-- nazev sloupce pro delku anomalie -->
<!ELEMENT timelength (#PCDATA)>
<!-- nazev sloupce pro zacatek anomalie v raw datech -->
<!ELEMENT indexrawstart (#PCDATA)>
<!-- nazev sloupce pro konec anomalie v raw datech -->
<!ELEMENT indexrawend (#PCDATA)>

<!ELEMENT imagewidget (path, prefix, extension)>
<!-- cesta do adresare s obrazky -->
<!ELEMENT path (#PCDATA)>
```

```
<!-- prefix nazvu obrazku -->
<!ELEMENT prefix (#PCDATA)>
<!-- pripona obrazku -->
<!ELEMENT extension (#PCDATA)>

<!ELEMENT videowidget (name, delayms, videopath)>
<!-- nazev widgetu -->
<!ELEMENT name (#PCDATA)>
<!-- zpozdeni videa pred signalem -->
<!ELEMENT delayms (#PCDATA)>
<!-- cesta k videosouboru -->
<!ELEMENT videopath (#PCDATA)>
]>
```

C.2 Konverze videa

Widget pro zobrazení videa umožňuje synchronizovat video soubory, ve kterých je možné vyhledávat. Dodaná videa byla ve formátu MOV, které tuto možnost nemají, tak je bylo nutné konvertovat do formátu AVI.

Ke konverzi byl použitý program `VirtualDub`, ve kterém byl jako výstupní formát zvolen soubor AVI a byl použitý kodek `Xvid`. Takto konvertovaný videosouboru již bylo možné připojit do synchronizace v aplikaci.

Příloha D

Obsah přiloženého CD

- `app` - zdrojové kódy aplikace
- `experiments`
  - `data` - vstupní/výstupí data pro experimenty
  - `docs` - dokumentace experimentů
  - `scripts` - zdrojové kódy jednotlivých experimentů
- `thesis` - text diplomové práce