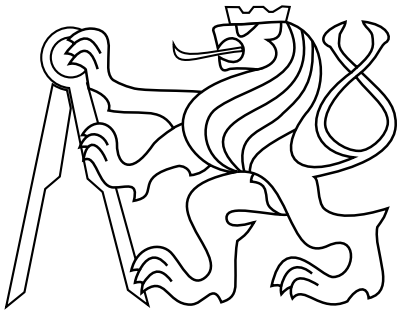
České Vysoké Učení Technické v Praze

Fakulta Elektrotechnická
Katedra kybernetiky
Robotické vnímání



Algoritmus více-objektového sledování a filtrování pózy v částečně pozorovatelném prostředí

Bakalářská práce

Frederik David Albl

Praha, Duben 2024

Studijní program: Kybernetika a robotika
Vedoucí práce: Mgr. Radoslav Škoviera, Ph.D.

Poděkování

V první řadě bych chtěl srdečně poděkovat svému vedoucímu práce Mgr. Radoslavu Škovierovi Ph.D., za odborné vedení, věnovaný čas a podporu při tvorbě práce. Také bych rád vyjádřil díky dalším pracovníkům z Českého institutu informatiky, robotiky a kybernetiky za poskytnutou pomoc při řešení potíží technického rázu a nakonec bych rád poděkoval svým blízkým, kteří mi byli během posledních měsíců velkou oporou.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Albl** Jméno: **Frederik David** Osobní číslo: **486061**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra kybernetiky**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Algoritmus více-objektového sledování a filtrování pózy v částečně pozorovatelném prostředí

Název bakalářské práce anglicky:

Multi-object Tracking and Pose Filtering Algorithm in Partially Observable Environment

Pokyny pro vypracování:

Cílem práce je implementace rychlého algoritmu filtrace pózy a víceobjektového sledování předmětů v částečně pozorovatelném prostředí. Algoritmus bude používat výstupy z již existujícího detektoru 6D pózy objektů. Řešení se musí vypořádat s přerušovanými a zašuměnými detekcemi, někdy s delšími výpadky detekce vznikajícími například vlivem okluzí objektů. Detekce mohou také přicházet z vícero kamer (tj. více detekcí stejného objektů ve stejném časovém okamžiku).
Úkoly:
1. Prozkoumejte vhodné algoritmy pro filtrování pózy objektu.
2. Implementujte a otestujte filtrování pózy pro jeden objekt za použití umělých dat.
3. Prozkoumejte přístupy víceobjektového sledování (MOT).
4. Zvolte a implementujte vhodný MOT algoritmus.
5. Zhodnoťte výkon provedení a přesnost implementovaného řešení na umělých a reálných datech.

Seznam doporučené literatury:

[1] Luo, Wenhan, et al. "Multiple object tracking: A literature review." Artificial intelligence 293 (2021): 103448.
[2] Dave, Achal, et al. "Tao: A large-scale benchmark for tracking any object." Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16. Springer International Publishing, 2020
[3] Roger Labbe. “Kalman and Bayesian Filters in Python” https://github.com/rlabbe/Kalmanand-Bayesian-Filters-in-Python
[4] Rakai, Lionel, et al. "Data association in multiple object tracking: A survey of recent techniques." Expert Systems with Applications 192 (2022): 116300.
[5] Liang, Tianyi, et al. "A generic MOT boosting framework by combining cues from SOT, tracklet and re-identification." Knowledge and Information Systems 63.8 (2021): 2109-2127.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Mgr. Radoslav Škoviera, Ph.D. robotické vnímání CIIRC

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **13.02.2024** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **21.09.2025**

Mgr. Radoslav Škoviera, Ph.D.	prof. Dr. Ing. Jan Kybic	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

Prohlášení

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

Dne..... .....

Abstract

This thesis explores the use of particle filtering for tracking and 6D pose estimation of multiple objects. The implemented algorithm should be working in real-time and it's supposed to be used for conveying information about the positions and orientations of objects on the scene for a robot, who is supposed to be manipulating them.

Three original variations of particle filter have been implemented. First, the filters have been analyzed and tested on the problem of tracking one object, i. e. *filtration*. Scripts allowing us to visualize and analyze the process of filtration were implemented in a visual simulator environment.

A system of generation, saving and uploading of saved trajectories based on user input was also implemented. After a first analysis of the filters' performance in visual environment, the filters have been systematically tested on generated data, which simulated noisy detector measurements.

Furthermore, methods of data association have been examined, specifically global nearest neighbour (GNN), probabilistic data association (PDA) and especially joint probability data association (JPDA). An algorithm based on JPDA has been designed and its basic implementation has been tested. The results show the algorithm needs further improvements.

Keywords multiple object tracking, object pose filtering, particle filter, Kalman filter, JPDA, GNN

Abstrakt

Tato práce se zabývá použitím částicového filtru pro sledování a odhad 6D pózy více objektů. Algoritmus má fungovat v reálném čase a sloužit ke zprostředkování odhadu pozic a orientací objektů na scéně robotovi, který s nimi má manipulovat.

Částicový filtr byl analyzován a implementován ve třech originálních provedeních. Nejprve byl otestován na problému sledování jednoho objektu, tedy filtraci. Ve vizuálním simulátoru byly vytvořeny skripty umožňující průběh filtrace sledovat, analyzovat a díky tomu i upravovat.

Byl vytvořen systém generování, ukládání a nahrávání dat o testovaných scénářích a trajektoriích dle zadaného uživatelského vstupu. Po prvotní analýze z vizuálního prostředí byly filtry systematicky testovány na vygenerovaných datech simulujících zašuměná měření detektoru. Zároveň byly otestovány různé kombinace parametrů, jejichž optimální hodnoty byly zaznamenány.

Dále byly v práci prozkoumány metody asociace dat, konkrétně GNN, PDA a zejména JPDA, navržen algoritmus založený na JPDA a otestována jeho základní implementace. Výsledky ukazují, že je potřeba algoritmus dále rozvinout.

Klíčová slova víceobjektové sledování, filtrace pózy objektu, částicový filtr, Kalmanův filtr, JPDA, GNN

Seznam zkratek

DBT sledování na základě detekcí *(z angl. detection based tracking)*

DFT sledování bez detekcí *(z angl. detection free tracking)*

EKF rozšířený Kalmanův filtr *(z angl. extended Kalman filter)*

GNN global nearest neighbour

JPDA joint probability data association

KPF částicový filtr s kalmanovým filtrem *(z angl. Kalman particle filter)*

LKF lineární Kalmanův filtr

MBRFS multi-Bernoulli Random finite set

MC Monte Carlo

MHT multiple hypothesis tracker

MOT více-objektové sledování (*z angl. multiple object tracking*)

MSE průměrná čtvercová chyba *(z angl. mean-squared-error)*

RMP Robotika a strojové vnímání *(z angl. Robotics and machine perception)*

PDA probabilistic data association

PF částicový filtr *(z angl. particle filter)*

PFGH částicový filtr g-h *(z angl. particle filter g-h)*

PMBM Poisson multi-Bernoulli mixture

PPP Poisson point process

RFS random finite set

SO směrodatná odchylka

Obsah

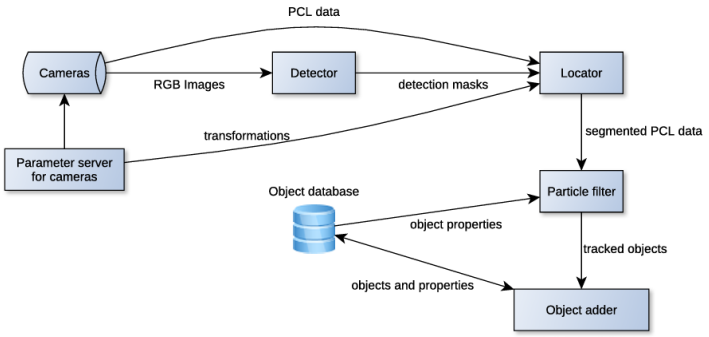
1 Úvod

Více-objektové sledování je významná úloha z oblasti robotiky a kybernetiky. Obecně řečeno jde o odhad pozice objektů ve scéně (např. v záběru kamery) pro jeden konkrétní časový okamžik a sledování trajektorie jednotlivých objektů v celém časovém rámci. Problém lze tedy zhruba rozdělit na 2 podproblémy. Za prvé jde o filtraci — tedy odhad trajektorie jednoho objektu ze zašuměných měření. Za druhé jde o asociaci dat — tedy správné přiřazování měření (a tím i odhadů pozice) k jednotlivým objektům.

Mnoho používaných algoritmů zahrnuje do řešení i samotnou detekci — způsob zpracování dat např. z obrazu k získání měřené hodnoty. Takovým přístupům říkáme end-to-end a často jsou řešeny pomocí neuronových sítí [1], [2], [4].

Nevýhodou tohoto přístupu je, že se neuronová síť naučí zpracovávat data pocházející pouze z určitého detektoru a nelze ji poté aplikovat pro detektor obecné povahy. Při oddělené tvorbě samotného algoritmu je naopak možné dosáhnout obecnějšího uplatnění, neboť zpracovávaná data mohou přicházet v různých formátech z různých detektorů. Díky tomu může být algoritmus aplikován na větší množství detektorů. Dále je takto možné algoritmus přizpůsobit danému problému v závislosti na požadované přesnosti, rychlosti a dalších parametrech úlohy.

Z tohoto důvodu se práce řešením skrz end-to-end neuronovou síť zabývat nebude. Výstup práce bude zakomponován do celkového systému počítačové vize zobrazeném na obr. 1.1, konkrétně částí označenou jako částicový filtr *(z angl. particle filter)* (PF). Předchozí části systému uvedené v diagramu na obr. 1.1 již má oddělení Robotika a strojové vnímání *(z angl. Robotics and machine perception)* (RMP) na CIIRC ČVUT hotové.



Obrázek 1.1

Cílem je vytvořit systém, který z přicházejících 6D dat o póze objektů (pozice a rotace) v čase bude schopen sledovat trajektorie objektů v prostoru. Konkrétně bude algoritmus použit do systému autonomních robotických manipulátorů pro správné určení pozice a pohybu objektů, aby tyto informace mohl robot využít.

Cílem algoritmu je, aby byl *rychlý*, *přesný* a *spolehlivý*. Pro vytvoření přesného algoritmu je potřeba vyšší výpočetní náročnost, proto je potřeba dojít mezi přesností a rychlostí ke kompromisu. V našem případě bude mít větší prioritu rychlost, aby algoritmus dobře fungoval

v reálném čase. Přesnost závisí spíše na vlastnostech detektoru, což není předmětem této práce. Za kritérium spolehlivosti můžeme považovat schopnost algoritmu:

(i) vypořádat se s přerušením detekcí objektu,
(ii) vypořádat se s okluzemi objektu,
(iii) vypořádat se s velmi nepřesnými měřeními,
(iv) vypořádat se s fantomovými detekcemi — když je objekt detekován tam, kde se zřejmě nenachází.

Úloha řešená v této práci má další specifika. V úloze bude znám počet objektů a jejich třídy. Přicházející data z detekčního systému v sobě rovněž budou obsahovat údaj o třídě detekovaného objektu. Dále bude systém komunikovat s robotem ve scéně, čímž může do svých odhadů zakomponovat např. údaj o uchopení nebo uvolnění objektu chapadlem robota. Pro účely filtrace bude použit částicový filtr a pro asociaci dat bude vyzkoušeno a použito vícero algoritmů uvedených níže v kap. 1.2.

1.1 Související práce

Algoritmy více-objektového sledování jsou podrobně popsány v přehledu literatury [12]. Autoři článku zkoumají různé klíčové komponenty v MOT systémech, rozdělují algoritmy dle kritérií (inicializace, mód zpracování, typ výstupu), předvádí výsledky experimentů a shrnují problémy výzkumu MOT. V závislosti na metodě inicializace objektu lze algoritmy rozdělit do 2 hlavních množin:

- sledování na základě detekcí *(z angl. detection based tracking)* (DBT) [16], [18],
- sledování bez detekcí *(z angl. detection free tracking)* (DFT) [13], [19].

V DBT jsou inicializace objektů prováděny na základě detekcí. Jsou tedy automatické a nedokonalé. Počet objektů je proměnlivý. V DFT se inicializují objekty manuálně a jejich počet je pevně daný. Výkon sledovače je tak nezávislý na detekci.

V našem případě máme sice pevně daný počet objektů, ale je třeba je inicializovat až ve chvíli, kdy se objeví na scéně. Naopak, když scénu opustí, je třeba jejich pozice z výstupních odhadů odstranit.

Článek [12] dále dělí algoritmy dle *modelu pohybu* na lineární a nelineární. Lineární model předpokládá stavovou rovnici ve tvaru:

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{Q} \tag{1.1}$$

, kde $\mathbf{x}_k$ je stav objektu v čase k, $\mathbf{Q}$ je procesní šum (např. gaussovský) a $\mathbf{F}$ je lineární přechodová rovnice, která mapuje změnu stavu objektu během jednoho časového kroku. Většina skutečného pohybu je nelineární, avšak za předpokladu lineárního pohybu se dají k výpočtům odhadu použít efektivní metody, jako např. lineární Kalmanův filtr (LKF). V našem případě bude problém nelinearity vyřešen vlastnostmi PF popsanými níže v kap. 2.1.2.

Algoritmy se také dělí dle způsobu zpracování dat na *online* a *offline* algoritmy. Offline algoritmy zpracovávají všechna naměřená data za celou dobu měření najednou a nelze je tedy použít v reálném čase. V našem případě je však výpočet v reálném čase nutností.

Dalším problémem při sledování více objektů je model interakce objektů, který se v praxi využívá například při sledování pohybu živočichů [22], nebo při modelování fyzických interakcí (srážky, vzájemná přitažlivost nebo odpudivost objektů atd.). Pro náš algoritmus

nebude uvažován žádný interakční model, neboť budeme pracovat s neživými objekty a fyzikální interakce objektů jsou v našem scénáři zanedbatelné, jejich modelování by tedy bylo nadbytečné.

Základní verze více-objektové sledování (*z angl. multiple object tracking*) (MOT) algoritmu je popsána v článku [25], jejíž struktura je popsána podrobněji v kap. 2.2. Této základní struktury se bude držet i námi vyvinutý algoritmus. Hlavní koncept spočívá v tom, že je nutné efektivně redukovat počet možných asociačních hypotéz, protože jejich množství velmi rychle (kombinatoricky) roste v každém kroku, dále je třeba vhodně odhadnout stav objektu na základě přijatých hypotéz.

Asi jeden z nejobecnějších algoritmů pro více-objektové sledování je tzv. Poisson multi-Bernoulli mixture (PMBM) [5], [10]. Tento filtr počítá s proměnlivým množstvím objektů a využívá konceptů jako je random finite set (RFS), Poisson point process (PPP) a multi-Bernoulli Random finite set (MBRFS), pomocí kterých lze vytvořit modely pro:

- objevování se objektů na scéně (*object birth*),
- mizení objektů ze scény (*object death*),
- falešná měření (*clutter*),
- množinu nedetekovaných objektů.

Algoritmus PMBM jednoznačně disponuje vlastnostmi řešícími naší úlohu. Naše úloha však obsahuje dodatečné informace, které její řešení usnadňují a lze ji tedy řešit jednodušším způsobem. Tyto informace jsou:

- znalost maximálního počtu objektů na scéně,
- znalost počtu objektů dle jednotlivých tříd.

Protože podmínkou řešení je vysoká rychlost algoritmu, byl by tedy PMBM filtr pro naše účely příliš výpočetně náročný a pro řešení úlohy nadbytečný.

Tři jednodušší varianty algoritmů jsou GNN [7], JPDA [15], [20] a multiple hypothesis tracker (MHT) [23]. GNN je nejjednodušší. Zvolí pouze hypotézu, která je nejpravděpodobnější a všechny ostatní nebere v úvahu. Výhodou je snadná implementace a rychlý výpočet, nevýhodou je vyloučení nejistoty a vyšší pravděpodobnost, že filtr na základě chybných měření přestane sledovat správný objekt.

JPDA algoritmus je založen na spojení (merge) všech hypotéz v každém kroku na základě jejich pravděpodobnosti. Tento algoritmus již lépe počítá s nejistotou, stále je výpočetně poměrně nenáročný, avšak funguje hůře v komplikovaných scénářích, například pokud jsou objekty blízko sebe [8].

MHT je z těchto tří algoritmů nejsložitější a nejobecnější. Sleduje určité množství asociačních hypotéz a každé přiřazuje váhu na základě její pravděpodobnosti [9]. V tomto algoritmu je zejména nutné správně redukovat množství nepravděpodobných hypotéz a efektivně počítat jejich váhy. Jelikož náš scénář není natolik komplikovaný a zásadním požadavkem je rychlost algoritmu, nabízí se zvolit algoritmus na bázi JPDA nebo GNN.

Výhodou sledování s PF, že PF v sobě může potenciálně obsahovat vícero hypotéz skrz rozložení částic [21]. Vzhledem k tomu, že je znám počet objektů na scéně a zároveň je pro filtraci použit PF, je možné, že dostatečně dobrá asociace dat půjde provést pomocí spíše jednodušího algoritmu.

V kombinaci s PF, což je metoda typu Monte Carlo (MC), se nabízí využít metody MC i pro asociaci dat, jako je tomu např. v tomto [17] článku, kde je asociace dat zároveň založena na metodě JPDA.

1.2 Přínos práce

Ve vizuálním prostředí simulátoru myGym byly vytvořeny programy sloužící k vizualizaci, analýze a ohodnocení filtrování 6D-pózy objektu a více-objektového sledování.

V první části práce zaměřené na samotnou filtraci jsou podrobně popsány teoretické základy problému, na nichž je založena implementace 3 různých originálních částicových filtrů, jejichž princip, vlastnosti a algoritmus jsou systematicky probrány v implementační sekci.

Součástí vytvořeného softwaru je skript umožňující generovat čtyři typy trajektorií dle zadaných parametrů, vizualizovat je, ukládat pro další použití, dále umožňuje použít tyto trajektorie pro zobrazení procesu filtrace. Byl vytvořen evaluační skript, který po zadání parametrů otestuje na uložených trajektoriích výkonnost jednotlivých filtrů.

Jednotlivé filtry byly systematicky testovány na vygenerovaných datech pro různé kombinace parametrů a jejich výsledky byly vyhodnoceny. Pro srovnání jejich účinnosti byl použit již implementovaný klasický Kalmanův filtr s přidanou funkcionalitou adaptivní filtrace. Byla zhodnocena pozitiva a negativa jednotlivých filtrů a pro další účely více-objektového sledování byl použit nejlepší z nich.

V druhé části práce byl popsán problém více-objektového sledování a byly analyzovány možnosti implementace jeho řešení za použití částicového filtru. Zmíněny byly některé algoritmy pro asociaci dat a podrobněji byly popsány 2 podobné algoritmy (PDA a JPDA), z jejichž základu byl nakonec implementován algoritmus použitý v této práci.

Volba právě těchto algoritmů byla zdůvodněna komplementárními vlastnostmi s částicovým filtrem. Tyto vlastnosti byly probrány v kapitole Teoretický rozbor a konkrétní implementace algoritmu byla popsána v kapitole 4. Byl rozšířen systém generování dat pro tvorbu scénářů s více trajektoriemi a jejich výsledný formát byl upraven pro evaluaci.

Systém vizualizace procesu filtrace byl rozšířen pro účely více-objektového sledování a byly provedeny prvotní experimenty testující účinnost sledovacích algoritmů GNN a JPDA

1.3 Matematické značení

$\mathbf{x}$, $\boldsymbol{\alpha}$	vektor
$\mathbf{X}, \boldsymbol{\Omega}$	matice
$\mathbf{I}$	jednotková matice
$x = \mathbf{a}^\intercal \mathbf{b}$	skalární součin $\mathbf{a}$, $\mathbf{b} \in \mathbb{R}^3$
$\mathbf{x} = \mathbf{a} \times \mathbf{b}$	vektorový součin $\mathbf{a}$, $\mathbf{b} \in \mathbb{R}^3$
$\mathbf{X}_{(a,b)}$	prvek matice (a, b)
$\dot{x}, \ddot{x}, \dddot{x}, \ddddot{x}$	první, druhá, třetí a čtvrtá časová derivace x
$\mathbf{x}_k$	stavový vektor $\mathbf{x}$ v kroku k
$\hat{\mathbf{x}}_k$	vektor odhadu stavu $\mathbf{x}$ v kroku k
$\mathbf{z}_{1:k}$	posloupnost měření $\mathbf{z}$ od kroku 1 ke kroku k
$\hat{\mathbf{x}}_{k\|k-1}$	vektor predikovaného odhadu stavu v kroku k pro měření $\mathbf{z}_{1:k-1}$
$\hat{\mathbf{x}}_{k\|k}$	vektor aktualizovaného odhadu stavu v kroku k pro měření $\mathbf{z}_{1:k}$
$\tilde{\mathbf{x}}_k^{(i)}, \tilde{\mathbf{x}}_k$	vektor vnitřního odhadu stavu v kroku k pro částici i nebo vnitřní stav Kalmanova filtru
$\bar{\mathbf{x}}_k$	předběžná hodnota vektoru x v kroku k
$\mathbf{F}, \mathbf{H}, \mathbf{Q}, \mathbf{R}$	matice: přechodového modelu, modelu pozorování, kovariance procesního šumu, kovariance pozorovaného šumu
$p(x), p(x\|z)$	pravděpodobnost a podmíněná pravděpodobnost
$\mathcal{N}(\mu, \sigma^2)$	normální rozdělení se střední hodnotou μ a rozptylem σ^2
$\mathbf{a} \equiv -\mathbf{a}$	deklarace hodnoty vektoru $\mathbf{a}$ na hodnotu $-\mathbf{a}$

Tabulka 1.1: Matematická notace, nomenklatura a vybrané symboly.

2 Teoretický rozbor

2.1 Filtrace

Nutnou součástí sledování více objektů je filtrace — tedy aproximace skutečné trajektorie objektu ze zašuměných měřených dat. K tomu se využívají rekurzivní algoritmy, které při každém kroku (při obdržení nového měření) s využitím reprezentací statistického šumu a jiných nepřesností, produkují odhad skutečného stavu objektu nebo systému.

Kalmanův filtr [11], [24] je základní algoritmus pro rekurzivní odhad stavu systému. Další algoritmy, které budeme v práci používat, z něj teoreticky vycházejí, a proto je potřeba si nejprve přiblížit fungování právě Kalmanova filtru.

Kalmanův filtr

Předpoklady

Kalmanův filtr (konkrétně *diskrétní Kalmanův filtr*) se snaží rekurzivně aproximovat skutečný stav systému nebo objektu $\mathbf{x}_k \in \mathbb{R}^n$ za následujících předpokladů:

- proces změny stavu objektu při každém kroku (přechodový model) se odehrává dle lineární stochastické diferenční rovnice [24] [1]

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{w} \tag{2.1}$$

.

- Měření $\mathbf{z}_k \in \mathbb{R}^m$ odpovídá modelu

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v} \tag{2.2}$$

.

- $\mathbf{w}$ a $\mathbf{v}$ jsou náhodné veličiny reprezentující procesní šum($\mathbf{w}$) a šum měření ($\mathbf{v}$). Předpokladem je, že jsou na sobě navzájem nezávislé, jedná se o bílý šum s normálním pravděpodobnostním rozdělením:

$$p(\mathbf{w}) \sim \mathcal{N}(0, \mathbf{Q}) \tag{2.3}$$
$$p(\mathbf{v}) \sim \mathcal{N}(0, \mathbf{R}) \tag{2.4}$$

- $\mathbf{F} \in \mathbb{R}^{n\times n}$ a $\mathbf{H} \in \mathbb{R}^{m\times n}$ jsou matice přechodového modelu a modelu pozorování.

V každé iteraci Kalmanova filtru se upraví:

- hodnota odhadovaného stavu $\hat{\mathbf{x}}_k$,
- kovariační matice stavu $\mathbf{P}_k$

Pravděpodobnostní rozdělení odhadovaného stavu je dáno:

$$p(\mathbf{x}_k|\mathbf{z}_k) = \mathcal{N}(\hat{\mathbf{x}}_k, \mathbf{P}_k) \tag{2.5}$$

[1] V obecném případě obsahuje rovnice ještě *řídící člen*, tedy $+\mathbf{B}\mathbf{u}_k$, ten byl ale vyřazen, neboť v našem případě nepracujeme s řídícím vstupem.

Rozbor

Teoretický základ konkrétních výpočtů v algoritmu Kalmanova filtru vychází *Bayesovského optimálního filtru* [6], [11]. Algoritmus Kalmanova filtru se provádí ve dvou zásadních krocích:

(i) **Predikce**: Výpočet $p(\mathbf{x}_k|\mathbf{z}_{1:k-1})$
(ii) **Aktualizace**: Výpočet $p(\mathbf{x}_k|\mathbf{z}_{1:k})$

Pro výpočet predikce je použita **Chapman-Kolmogorovova rovnice**:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})d\mathbf{x}_{k-1} \tag{2.6}$$

A pro výpočet aktualizace se vychází z **Bayesova pravidla**:

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k, z_{1:k-1})p(x_k|z_{1:k-1})}{p(z_k)} \tag{2.7}$$

Rovnice 2.6 a 2.7 jsou matematickým základem pro všechny filtry odvozené od Kalmanova. To platí pro rozšířený Kalmanův filtr *(z angl. extended Kalman filter)* (EKF) nebo PF používanými v této práci. Rovnice Kalmanova filtru (viz. kap. 2.1.1) jsou přímo matematicky odvoditelné z rovnic (2.6) a (2.7), zatímco například PF z nich pouze přibližně vychází. Kalmanův filtr je zároveň *optimálním filtrem* ve smyslu *nejmenších čtverců* pro trajektorie a měření splňující výše uvedené (kap. 2.1.1) předpoklady.

Algoritmus Kalmanova filtru

Jak již bylo řečeno, algoritmus sestává z dvou kroků.

(i) Predikce:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\hat{\mathbf{x}}_{k-1|k-1} \tag{2.8}$$
$$\mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^T + \mathbf{Q} \tag{2.9}$$

,kde $\hat{\mathbf{x}}_{k|k-1}$ a $\hat{\mathbf{P}}_{k|k-1}$ jsou tzv. *prior* hodnotami x a $\mathbf{P}$, tedy predikované hodnoty před zakomponováním měření, odhadované ze stavu v předchozím kroku.

(ii) Aktualizace:

$$\mathbf{S}_k = \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R} \tag{2.10}$$
$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}\hat{\mathbf{x}}_{k|k-1}\mathbf{S}_k^{-1} \tag{2.11}$$
$$\mathbf{y}_k = \mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1} \tag{2.12}$$
$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\mathbf{y}_k \tag{2.13}$$
$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_{k|k-1} \tag{2.14}$$

,kde $\mathbf{S}_k$ je matice inovační kovariance, $\mathbf{K}_k$ je tzv. Kalmanův zisk, $\mathbf{y}_k$ je reziduum (rozdíl mezi skutečným měřením $\mathbf{z}_k$ a predikovaným měřením $\mathbf{H}_k\hat{x}_{k|k-1}$) a $\hat{\mathbf{x}}_{k|k}$ a $\hat{\mathbf{P}}_{k|k}$ jsou odhady stavu a kovarianční matice stavu již pro další krok.

Samotný algoritmus pak sestává z inicializace $\hat{\mathbf{x}}_1$ a $\hat{\mathbf{P}}_1$ a následného opakování kroků predikce a aktualizace při každém měření. Více v kap. 3.3.

Částicový filtr

Během filtrace a sledování objektu vyžadujeme po filtru několik vlastností [11]:

- **Multimodálnost** — chceme sledovat najednou více hypotéz o objektu.
- **Okluze** — chceme, aby se filtr dokázal vypořádat s okluzemi objektu.
- **Nelineární chování** — chceme, aby filtr reagoval rychle na nelineární změny v chování, popřípadě na chybné detekce.
- **Robustní přechodový model** — filtr by měl být robustní i vůči poměrně velkým změnám pohybu.

Tyto požadavky velice dobře splňuje PF. Jeho princip je, že modeluje pravděpodobnostní rozdělení odhadovaného stavu objektu $\hat{\mathbf{x}}_k$ nikoliv pomocí normálního rozdělení, nýbrž konečným počtem *částic*. Částice by měly reprezentovat vzorky vygenerované ze skutečného pravděpodobnostního rozdělení odhadu stavu objektu a jejich hustota přibližně modeluje pravděpodobnost výskytu objektu na dané pozici.

Jednotlivé kroky algoritmu, tedy **predikce** a **aktualizace**, se realizují na každou částici zvlášť. To umožňuje dosáhnout výše zmíněných vlastností:

- Částice mohou být libovolně rozdělené a pokud jich je dostatečný počet, mohou v sobě obsahovat vícero různých hypotéz.
- Mezi tyto hypotézy může patřit i okludovaný objekt.
- Výpočet predikce a aktualizace nemusí být lineární a lze aplikovat na každou částici zvlášť.
- Variabilita částic umožňuje lepší adaptaci na obecný model pohybu objektu.

Obecný algoritmus

Obecný algoritmus pro částicový filtr lze popsat v pěti krocích [11]:

(i) **Vygenerování částic**
Částice mohou mít pozici, rychlost nebo jakoukoliv jinou stavovou veličinu, kterou je třeba odhadovat. Každá částice má svou váhu, která odpovídá tomu, jak moc se její stav shoduje se skutečným stavem objektu.

(ii) **Predikce**
Přesun částic dle přechodového modelu objektu.

(iii) **Aktualizace**
Aktualizace vah částic dle získaného měření. Částice blíže k měření mají vyšší váhu než částice, které jsou měření vzdálené.

(iv) **Odhad stavu**
Stav se odhadne na základě váženého průměru částic.

(v) **Převzorkování**
Zbavení se velmi nepravděpodoboných částic a jejich nahrazení kopiemi pravděpodobnějších částic.

V dalších iteracích se opakují kroky (ii)–(v).

Tento obecný algoritmus lze implementovat mnoha způsoby. V této práci byly vytvořeny 3 různé algoritmy na bázi částicového filtru a jejich výkony byly otestovány a vzájemně srovnány (kap. 3.4, kap. 5.1)

Struktura implementace

Samostatné řešení úlohy filtrace spočívalo v následujících podúkolech:

(i) Vizualizace
  (a) spuštění vizuálního prostředí
  (b) načtení a správné zpracování dat o trajektorii
  (c) vizualizace trajektorie
  (d) vizualizace pohybu a orientace předmětu po trajektorii
  (e) vizualizace částic filtru a výstupu filtru (odhad pozice)
  (f) animace procesu filtrace (pohyb částic, předmětu, odhadu pozice)

(ii) Implementace filtru
  (a) analýza problému pro tvorbu vhodného filtru a algoritmus filtrace
  (b) vytvoření třídy s parametry a metodami užitými pro filtraci
  (c) zakomponování algoritmu filtrace do vizualizace
  (d) určení vhodných (a realistických) parametrů filtrace a šumu dat

(iii) Zpracování výsledků
  (a) zajištění správného ukládání výstupu filtru
  (b) vygenerování vhodného souboru dat
  (c) systematické testování filtrů na větším souboru dat
  (d) hledání optimálních parametrů pro jednotlivé filtry.

2.2 Víceobjektové sledování

Na úlohu filtrace navazuje víceobjektové sledování. Také jde o rekurzivní filtraci zašuměných měření, které probíhá ve stejných hlavních krocích jako u filtrace samotné — predikce a aktualizace. Základní rozdíl oproti úloze filtrace spočívá v tom, že se na scéně vyskytuje současně více objektů a v každém kroku filtru se vyskytne více měření. Data pocházející z detektoru tedy přichází z různých míst na scéně a je potřeba správně určit, která měření přísluší danému objektu — tento problém se nazývá **asociace dat**.

Asociace dat

Obecně lze problém asociace dat formulovat následovně: každou z naměřených hodnot množiny $\cup_{i=1}^{m}\{\mathbf{z}_k, j\}$ přiřadíme jednomu ze sledovaných objektů, jejichž stav $\mathbf{x}_{k,i}$ odhadujeme.

Tímto způsobem je možné pro každý sledovaný objekt (ten je reprezentován pravděpodobnostním rozdělením svého odhadu, v našem případě částicemi filtru a jejich vahami) provést krok aktualizace s měřením, které k němu bylo asociováno.

Aby byl filtr přesný, je potřeba zvážit nejistotu v přicházejících naměřených hodnotách. Naměřená hodnota může odpovídat více objektům nebo to může být hodnota neodpovídající žádnému objektu. Je potřeba pracovat s větším množstvím *asociačních hypotéz*, tedy sekvencí asociací v každém kroku.

Problémem při asociaci dat je rychle rostoucí množství asociačních hypotéz. Asociační hypotézu pro krok k lze definovat jako:

$$\theta_k = \begin{bmatrix}\theta_{k,1} & \dots & \theta_{k,m}\end{bmatrix} \tag{2.15}$$

kde m je počet měření, a $\theta_{k,1}$ až $\theta_{k,m}$ nabývají hodnot $[0 \quad \dots \quad n]$. Hodnota $\theta_{k,i}$ a index i u $\theta_{k,i}$ určují, že měření $\mathbf{z}_{k,i}$ bude asociováno objektu s indexem $\theta_{k,i}$. Hodnota $\theta_{k,i} = 0$ znamená, že se jedná o falešné měření a není přiřazené žádnému objektu.

V každém kroku získáme celkem $\binom{n}{m+1}$ nových asociačních hypotéz. Celkové množství hypotéz tak kombinatoricky roste, protože každou z hypotéz v novém kroku můžeme přiřadit ke každé z řady asociačních hypotéz z kroků minulých.

Není možné počítat se všemi možnými asociačními hypotézami a je nutné jejich množství snížit, aniž by tím mnoho utrpěla přesnost sledování.

Dva základní způsoby snižování počtu hypotéz jsou:

- ořezávání,
- spojování.

Ořezáváním se jednoduše vyřadí některé méně pravděpodobné asociační hypotézy v každém kroku. Příkladem algoritmu, který využívá ořezávání hypotéz je GNN. Tento algoritmus přijímá pouze jednu nejpravděpodobnější asociační hypotézu v každém kroku.

Spojování redukuje počet asociačních hypotéz tím, že se několik z nich zkombinuje dohromady. Základní algoritmus, který pro redukci množství asociačních hypotéz používá spojování, je PDA nebo JPDA.

Pro naše účely byla použita právě metoda JPDA, neboť se dobře doplňuje s částicovým filtrem. Algoritmus JPDA vychází z algoritmu PDA, který je popsán v následující kapitole.

Probabilistic data association (PDA)

PDA pro jeden objekt

Algoritmus JPDA je rozšířením jeho zjednodušené verze PDA, která se používá při sledování pouze jednoho objektu. Základní ideou algoritmu PDA je spojení všech asociačních hypotéz do jedné. Výsledný odhad o stavech objektu je poté dán jedním pravděpodobnostním rozdělením aproximujícím sadu pravděpodobnostních rozdělení z každé možné hypotézy.

Pokud by odhad stavu objektu byl reprezentován přesným pravděpodobnostním rozdělením zahrnujícím všechny asociační hypotézy od prvního kroku filtru, mohli bychom ho reprezentovat jako:

$$p(\mathbf{x}_k) = \sum_{i=1}^{\vartheta_{1:k}} w_i \cdot p(\mathbf{x}_k)_i \tag{2.16}$$

kde $\vartheta_{1:k}$ je množina všech asociačních hypotéz od prvního kroku do kroku k, w_i je váha dané hypotézy a $p(\mathbf{x}_{k,i}) \sim \mathcal{N}(\mu_i, \mathbf{P}_i)$ je pravděpodobnostní rozdělení stavu objektu pro danou asociační hypotézu. Algoritmus PDA aproximuje toto rozdělení jedním Gaussiánem $p(\mathbf{x}_k)^{PDA} \sim \mathcal{N}(\mu, \sigma^2_{PDA})$, které má stejnou střední hodnotu a rozptyl, jako má rozdělení přesné.

Tato aproximace poskytuje jednoduchost výpočtů s pouze jednou hypotézou stejně jako algoritmu GNN, ale narozdíl od něj si uchovává informace o ostatních možných asociacích.

Algoritmus PDA se uplatňuje ve fázi aktualizace:

- Pro objekt máme predikované pravděpodobnostní rozdělení odhadu stavu $p(\mathbf{x}_{k|k-1})$.
- V kroku k získáme m měření, máme tedy celkem $m+1$ asociačních hypotéz θ_k, kde $\theta_k = i$ určuje index měření $\mathbf{z}_{k,j}$, které je asociováno objektu
- Pokud je hodnota θ_k rovna nule, jsou všechna měření považována za "nepořádek"(*z angl. "clutter"*)
- Pro každou z asociačních hypotéz θ_k lze spočítat posteriorní distribuci $p(\mathbf{x}_k|\mathbf{z}_{k,j}) \sim \mathcal{N}(\boldsymbol{\mu}_i, \mathbf{P}_{k|k,i})$ standardním způsobem dle použitého filtru.

- Výsledná posteriorní distribuce $p(\mathbf{x}_{k|k})$ bude dána Gaussovským rozdělením, jehož střední hodnota je váženým průměrem středních hodnot posteriorních rozdělení získaných z asociačních hypotéz [6]:

$$\mathbb{E}[p(\mathbf{x}_{k|k})] = \sum_{i=1}^{m+1} w_i \boldsymbol{\mu}_i = \boldsymbol{\mu}_{k|k} \tag{2.17}$$

- Váhy w_i ve vztahu 2.17 jsou úměrné pravděpodobnosti (z angl. *likelihood*) $p(\mathbf{z}_{k,j}|\mathbf{x}_{k|k-1})$ a pro jejich přesné určení je stačí normalizovat jejich sumou
- Rozptyl (kovarianční matici) posteriorního rozdělení $p(\mathbf{x}_{k|k}) \sim \mathcal{N}(\boldsymbol{\mu}_i, \mathbf{P}_{k|k})$ lze spočítat dle vzorce

$$\mathbf{P_{k|k}} = \sum_{i=1}^{m} w_i \mathbf{P}_{k|k,i} + w_i(\boldsymbol{\mu}_{k|k} - \boldsymbol{\mu}_i)(\boldsymbol{\mu}_{k|k} - \boldsymbol{\mu})^T \tag{2.18}$$

Poměrně jednoduchý algoritmus PDA bohužel však nelze sám o sobě využít pro sledování více objektů. Je potřeba modelovat odhad pro každý objekt a ty na sobě nejsou nezávislé - při zjišťování pravděpodobnosti (likelihood) naměřené hodnoty za předpokladu predikovaného rozdělení stavu filtru je potřeba zohlednit ostatní objekty.

Pro tyto účely byl vyvinut algoritmus JPDA, který je podrobněji popsán v následující kap. 2.2.3.

Joint probabilistic data association JPDA

Algoritmus JPDA funguje v základu stejně jako PDA. Důležitým rozdílem je jeho schopnost sledovat i více objektů. S více objekty roste počet asociačních hypotéz ještě rychleji a spojování hypotéz do jedné přináší obrovské snížení výpočetní náročnosti.

Hlavní rozdíl oproti PDA je počítání vah w_i – pravděpodobnosti $p(\mathbf{z}_{k,\theta_k}|\mathbf{x}_{k|k-1;i})$ je třeba počítat **sdruženě**. To znamená, že pravděpodobnost přiřazení měření $\mathbf{z}_{k,i}$ k objektu $\mathbf{x}_{k,j}$ nezávisí pouze na jeho "blízkosti"k pravděpodobnostnímu rozdělení objektu, nýbrž také na jeho blízkosti k *ostatním* objektům.

Výsledná distribuce objektu j bude opět váženým průměrem posteriorních rozdělení získaných za podmínky určité asociační hypotézy:

$$p(\mathbf{x}_{k|k,i}) = \sum_{j=1}^{m} \beta_{i,j} \cdot p(\mathbf{x}_{k|k,i}|\mathbf{z}_{k,j}) + \beta_{j,0} \cdot p(\mathbf{x}_{k|k-1,i}) \tag{2.19}$$

kde $\beta_{i,j}$ je takzvaná *marginální asociační pravděpodobnost* pro přiřazení měření i objektu j a $\beta_{0,j}$ je marginální asociační pravděpodobnost pro nepřiřazení žádného měření k objektu i. Hodnota $\beta_{i,j}$ je určována nenormalizovanými vahami $\tilde{w}_{i,j}$, které se vypočtou jakožto pravděpodobnost (*likelihood*) naměření hodnoty $\mathbf{z}_{k,j}$ pro daný objekt $\hat{\mathbf{x}}_{k,i}$.

Hodnoty $\tilde{w}_{i,j}$ jsou rovny funkčním hodnotám celkem m Gaussovských rozdělení

$$\mathcal{N}(\mathbf{z}_{k,j}, \mathbf{R}) \tag{2.20}$$

v bodě $\hat{\mathbf{x}}_{k|k-1,j}$, kde

$$R = \begin{bmatrix} \sigma_z^2 & 0 & 0 \\ 0 & \sigma_z^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{bmatrix} \tag{2.21}$$

je kovarianční matice šumu měření.

Tyto váhy můžeme vypsat do tabulky 2.22 (fiktivní příklad pro 3 objekty a 4 měření v kroku k):

$\mathbf{x}_{k,i} \backslash \mathbf{z}_{k,j}$	1	2	3	4	0
1	$\tilde{w}_{1,1}$	$\tilde{w}_{1,2}$	$\tilde{w}_{1,3}$	$\tilde{w}_{1,4}$	$\tilde{w}_{1,0}$
2	$\tilde{w}_{2,1}$	$\tilde{w}_{2,2}$	$\tilde{w}_{2,3}$	$\tilde{w}_{2,4}$	$\tilde{w}_{2,0}$
3	$\tilde{w}_{3,1}$	$\tilde{w}_{3,2}$	$\tilde{w}_{3,3}$	$\tilde{w}_{3,4}$	$\tilde{w}_{3,0}$

(2.22)

Máme celkem 28 asociačních hypotéz a 15 nenormalizovaných vah $\tilde{w}_{i,j}$, z kterých lze vypočíst marginální asociační pravděpodobnosti $\beta_{i,j}$. Lze ji vypočíst tak, že pro všechny hypotézy $\mathbf{\Theta}_k \in$ $_\mathbf{k}$, kde je objektu i přiřazeno měření j, sečteme všechny nenormalizované váhy $\tilde{w}_{i,j}$ asociací měření k objektům a poměříme tuto sumu se sumou vah pro všechny asociační hypotézy.

Vyhotovíme tabulku sdružených vah (nenormalizované pravděpodobnosti), kde pro každou hypotézu (sloupce) a objekt (řádky) vypíšeme váhu $\tilde{w}_{i,j}$, která pro danou asociaci měření k objektu vyjadřuje pravděpodobnost $p(\mathbf{z}_{k,j}|\hat{\mathbf{x}}_{k|k-1,i})$:

Objekt j \ θ_k	$\begin{bmatrix}1 & 2 & 3\end{bmatrix}$	$\begin{bmatrix}1 & 3 & 2\end{bmatrix}$	...	$\begin{bmatrix}1 & 0 & 0\end{bmatrix}$	...
1	$\tilde{w}_{1,1}$	$\tilde{w}_{1,1}$	...	$\tilde{w}_{1,1}$	...
2	$\tilde{w}_{2,2}$	$\tilde{w}_{2,3}$	...	$\tilde{w}_{2,0}$	...
3	$\tilde{w}_{3,3}$	$\tilde{w}_{3,2}$	...	$\tilde{w}_{3,0}$	...

Tabulka 2.1: Příkladová tabulka nenormalizovaných vah

Nyní můžeme spočíst marginální asociační pravděpodobnost $\beta_{1,1}$:

$$\beta_{1,1} = \frac{\sum_{\theta_k \in \vartheta_k, \theta_k[1]=1, j\neq 0} \tilde{w}_{i,j;i,j\in\theta_k};}{\sum_{\theta_k \in \vartheta_k} \tilde{w}_{i,j;i,j\in\theta_k}} \tag{2.23}$$

Analogicky by se spočetly ostatní marginální asociační pravděpodobnosti. Tento výpočet je však náročný a počet hypotéz v každém kroku narůstá, a tak se provádí aproximační metody, které snižují množství hypotéz a tedy i nenormalizovaných vah zahrnutých do výpočtu posteriorního rozdělení.

Algoritmus JPDA poskytuje precizní výpočet aproximace posteriorního rozdělení odhadu stavu $\mathbf{x}$, který zahrnuje vliv každého měření, a je tak robustnější vůči měřícím chybám než filtr GNN. V této práci nebyla metoda JPDA sice využita přímo, ale její podstatné prvky byly použity k řešení asociace dat ve víceobjektovém sledování v kombinaci s částicovým filtrem. Následující kap. 2.2.4 popisuje vzájemné fungování principů z algoritmů PDA a JPDA s částicovým filtrem a jakým způsobem se jejich vlastnosti doplňují.

■ PDA v kombinaci s částicovým filtrem

Doplňující se vlastnosti

Jak již bylo řečeno, algoritmus PDA, případně JPDA disponuje vyšší rychlostí a zahrnutím všech měřených hodnot, na druhou stranu však trpí nespecifičností odhadu stavu a může tak ztratit pojem o významnějších sekvencích asociačních hypotéz, čímž může dojít ke zhoršení přesnosti sledování. Další problém algoritmu JPDA je narůstající počet asociačních hypotéz, které je třeba periodicky spojovat nebo jejich počet ořezávat.

Naproti tomu částicový filtr disponuje vlastnostmi, které jsou k vlastnostem asociace dat ve stylu JPDA komplementární:

- **Částicový filtr může přirozeně obsahovat více hypotéz** — poskytuje díky své *multimodálnosti* větší specifičnost hypotéz (každá částice může být považována za jakousi "mini-hypotézu") a lze tedy držet jejich rozdělením údaj o vícero hypotézách o skutečné pozici objektu, aniž by se tím zvyšovala výpočtová či paměťová náročnost vzhledem k základnímu fungování filtru
- Vysoké množství částic zajišťuje automatický **implicitní výpočet marginálních asociačních pravděpodobností** $\beta_{i,j}$ — tyto pravděpodobnosti tedy není nutné explicitně počítat a algoritmus se tak podstatně zjednoduší

V kroku aktualizace tedy stačí jednotlivým částicím připsat nenormalizované váhy $\tilde{w}_{i,j}$ na základě $p(\mathbf{z}_{k,j}|\hat{\mathbf{x}}_{k|k-1,i})$ pro všechna měření, což je podobné algoritmu PDA. Pravděpodobnost samotné marginální asociace $\beta_{i,j}$ se implicitně projeví v množství částic, které jsou daným měřením silněji ovlivněny (byla jim připsána vyšší nenormalizovaná váha $\tilde{w}_{i,j}$), oproti jiným filtrům, jejichž částice se tak blízko danému měření nenachází).

Celkové vlastnosti algoritmu

Díky těmto komplementárním vlastnostem lze vytvořit algoritmus víceobjektového sledování s požadovanými vlastnostmi i nižší výpočtovou náročností, což je vzhledem ke snaze o výpočet v reálném čase žádoucí. Další výhodou použití částicového filtru je, že dodatečné informace o scéně (například počet objektů z jednotlivých tříd na scéně, informace z detektoru o pravděpodobné třídě objektu, jemuž přísluší detekovaného měření) mohou být jednoduše zahrnuty do algoritmu tak, že se upraví nenormalizované váhy částic.

Shrnutí vlastností použitého algoritmu (jeho podrobná struktura a výpočty v jednotlivých krocích popsány v kap. 4):

- Může zahrnovat vyšší a proměnlivé množství asociačních hypotéz díky většímu počtu částic a jejich stochastickému chování
- Do asociace dat jsou zahrnuta všechna měření a tím vypadává riziko spojené s odříznutím hypotézy, která se nakonec ukáže jako pravdivá, které nastává u algoritmu GNN
- Náročnost výpočtu oproti samotnému částicovému filtru se pouze násobí počtem objektů a počtem měření v kroku k, není třeba explicitně zahrnovat všechny kombinace asociačních hypotéz Toto je vlastností algoritmu PDA, který není výpočetně náročný
- Částicová reprezentace pravděpodobnostního rozdělení umožňuje jednoduché zahrnutí dalších informací o scéně do algoritmu tím, že se různými způsoby upraví nenormalizované váhy jednotlivých částic

Struktura implementace

Řešení úlohy víceobjektového sledování lze rozdělit do následujících podúkolů podrobněji popsaných v kap. 4:

(i) Vizualizace
  (a) Generování scénářů s více trajektoriemi
  (b) Změna formátování dat — jejich přizpůsobení řešení víceobjektového sledování
  (c) Rozšíření zpracování uživatelského vstupu pro zahrnutí víceobjektového sledování
  (d) Simultánní zobrazení více trajektorií, měření, odhadů a částic

(ii) Implementace algoritmu
- Analýza problému pro zvolení vhodného přístupu implementace algoritmu
- Návrh algoritmu
- Vytvoření příslušných funkcí u tříd jednotlivých filtrů k umožnění víceobjektového sledování
- Zakomponování algoritmu do vizualizace

(iii) Zpracování výsledků

(iv) Otestování vizuálního scénáře

(v) Zpracování výsledků, které dokazují koncept fungování filtru

3 Implementace filtrace

V práci byly vytvořeny 4 různé filtrovací algoritmy na odhadu 6D pózy jednoho objektu. Jejich implementace je popsána v této kapitole a v kap. 5.1 jsou rozebrány jejich výsledky.

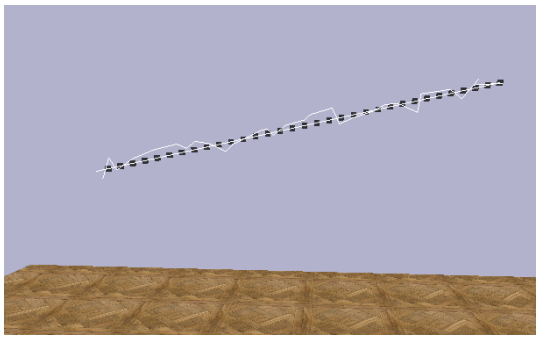
3.1 Rozbor problému

Pro lepší vyzkoušení funkčnosti a srovnání výkonu filtrů byl vytvořen zjednodušený scénář, na kterém budou filtry testovány.

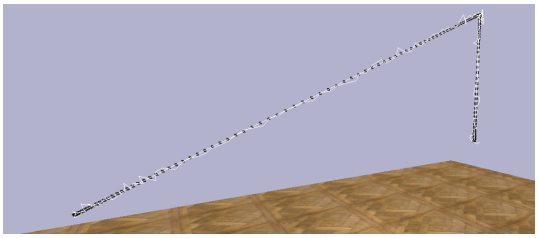
Parametry zadání

Filtr má být otestován na čtyřech typech trajektorií:

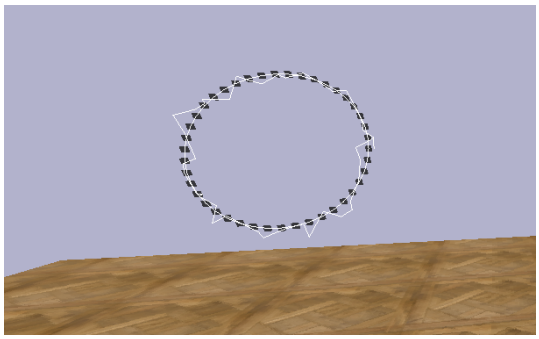
(i) Posloupnost úseček s konstantními rychlostmi
(ii) Posloupnost úseček s konstantními zrychleními
(iii) Kružnice s konstantní velikostí rychlosti
(iv) Složitější prostorová křivka (např. spline) s proměnlivou rychlostí i zrychlením



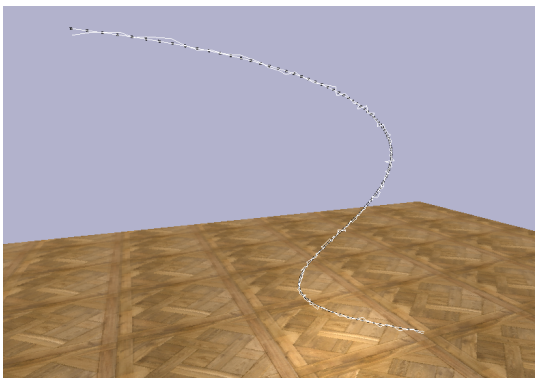
(a) Ukázka vygenerované úsečky



(b) Ukázka vygenerovaných úseček se zrychlením



(c) Ukázka vygenerované kružnice



(d) Ukázka vygenerovaného splinu

Obrázek 3.1: Ukázky vygenerovaných trajektorií

Tyto trajektorie představují "pravdivá"(ve skutečnosti neznámá) vstupní data o 6D póze objektu. K těmto datům byl uměle přidán Gaussovský šum s nulovou střední hodnotou a konstantní směrodatnou odchylkou. Zašuměná data pak představovala simulaci "skutečných"dat přicházejících z kamery (detektoru).

Schopnost filtru aproximovat orientaci objektu bude otestována na lineárně extrapolované rotační trajektorii mezi klíčovými body trajektorie (například u posloupnosti úseček to jsou body oddělující sousední úsečky).

Protože se dají orientace a pozice objektu považovat za vzájemně *nezávislé*, lze jejich filtraci od sebe navzájem oddělit. Stavový vektor $\mathbf{x}$ bude tedy obsahovat pouze souřadnice pozice objektu.

Vstupní data pro pozici objektu tedy představují pro každý krok k vektor :

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & z_k \end{bmatrix}^T \tag{3.1}$$

Reprezentací orientace objektu je mnoho a lze je mezi sebou bez problémů převádět. Filtry v této práci budou počítat s **kvaternionovou** reprezentací z důvodu stabilních výpočetních operací. Obecný kvaternion lze zapsat ve formátu $q = a + bi + cj + dk$, kde i, j, k jsou komplexní členy kvaternionu.

Vstupní data pro orientaci objektu tedy představují pro každý krok k vektor :

$$\mathbf{q}_k = \begin{bmatrix} a_k & b_k & c_k & d_k \end{bmatrix}^T \tag{3.2}$$

3.2 Způsob testování a vizualizace

Aby bylo možné filtry otestovat, bylo potřeba vytvořit nebo použít jednotné rozhraní pro inicializaci trajektorie, filtru a následné provedení samotné filtrace. K tomu byl využit simulátor MyGym [3] skupiny Incognite působící na CIIRC ČVUT v Praze. Simulátor je určen k rychlému prototypování neuronových sítí v oblasti robotické manipulace a navigování.

Zároveň simulátor poskytuje vizuální prostředí (s pomocí knihovny *pybullet*), které umí zobrazit trajektorii objektu, samotný objekt, i částice filtru. To bylo využito pro analýzu funkčnosti tvorby trajektorií i samotných filtrů.

V následujících kapitolách se budeme věnovat fungování skriptu *filter_test.py*, což je hlavní skript sloužící k vizualizaci filtrace a generování trajektorií.

Spuštění vizuálního prostředí

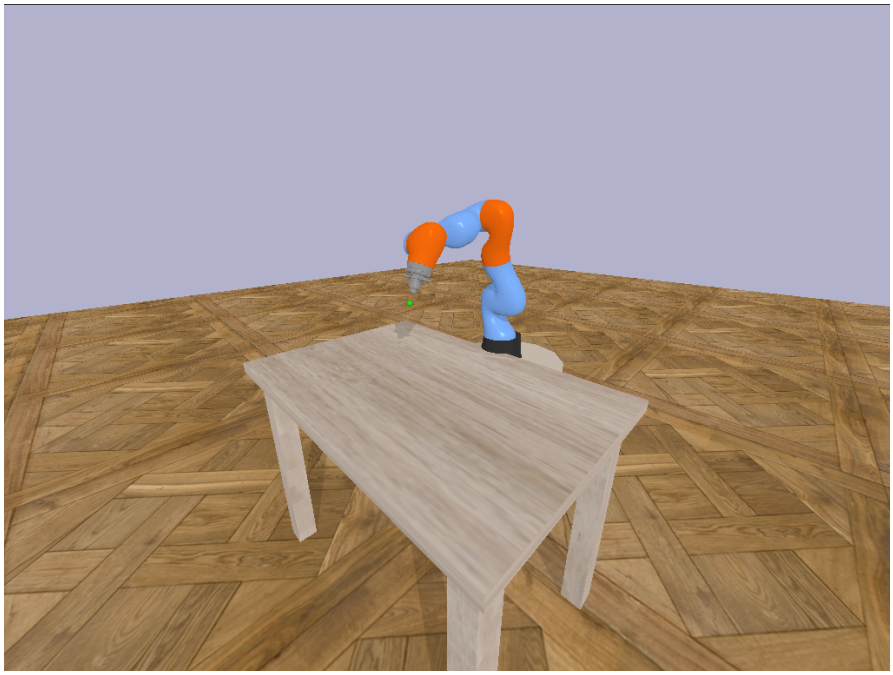
Spouštěcí skript *filter_test.py* byl vytvořen z části podle již existujícího skriptu *test.py*, který slouží k testování robotických úloh ve vizuálním prostředí. Bylo inicializováno prostředí, jak lze vidět na obr. 3.2. Bylo nutné poupravit některé parametry jiných souborů (bez zásahu do ostatních funkcí simulátoru), aby vizualizaci nepřekážely funkcionality sloužící testování robotických úloh.

Načtení a zpracování uživatelského vstupu

Po spuštění program čeká na uživatelský vstup. Uživatel zadá, zda chce generovat trajektorii, nebo zobrazit proces filtrace.

Generování trajektorie

Pokud si uživatel zvolí generování trajektorie, je vyzván ke zvolení jednoho ze čtyř typů trajektorií (viz. kap. 3.1.1). Poté se spustí vizuální prostředí a vygeneruje se jedna trajektorie zvoleného typu (popis generování trajektorií je podrobně popsán v kap. 3.9.1).



Obrázek 3.2: Ukázka pybullet prostředí mygym

Dále uživatel pomocí zadání "1"tuto trajektorii uloží do předem definované složky, nebo pomocí zadání "0"se trajektorie neuloží. Následně se vygeneruje nová trajektorie a proces se opakuje. Tento proces slouží ke generování vhodné sady dat, která se může následně použít k evaluaci filtrace.

Vizualizace procesu filtrace

Pokud si uživatel zvolí vizualizaci procesu filtrace, musí dále určit, zda chce vizualizaci provést na již vygenerované trajektorii ze souboru dat, nebo jestli si chce nechat vygenerovat vlastní.

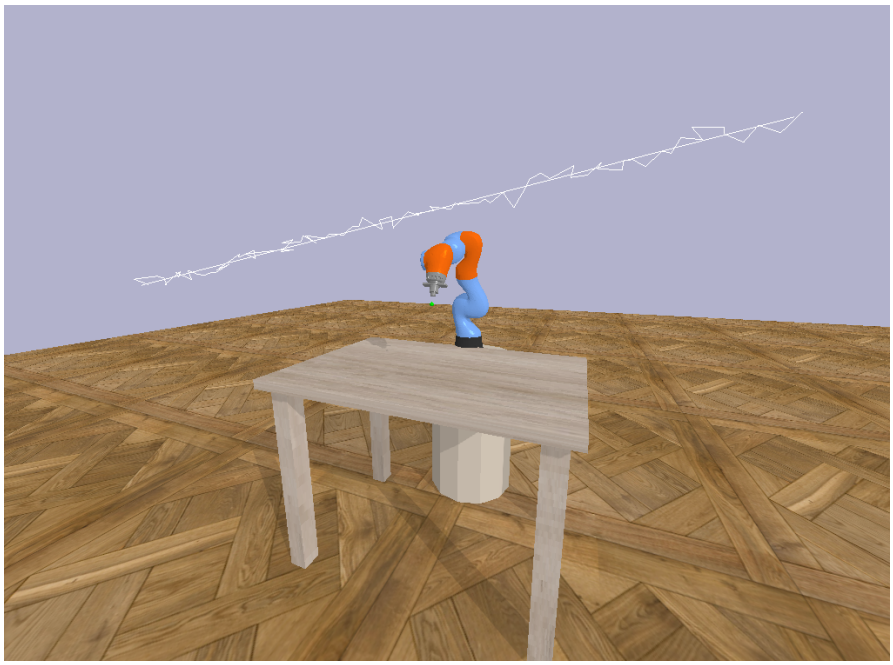
Jestliže si vybere trajektorii již vytvořenou, nejprve si zvolí její typ a poté se načte trajektorie zvoleného typu ze sady dat. Poté si zvolí jeden ze 4 implementovaných typů filtru, který bude použit pro filtraci (filtry popsány v kapitolách 3.3, 3.5, 3.7 a 3.6). Filtr je inicializován s parametry uvedenými v souboru *filter_parameters.json*.

Jestliže si uživatel vybere možnost generování trajektorie, zvolí si typ trajektorie a poté se generují trajektorie způsobem popsaným v kap. 3.2.2. Uživatel po zadání "1"zvolí zobrazenou trajektorii a dále si zvolí typ filtru, který bude použit k filtraci.

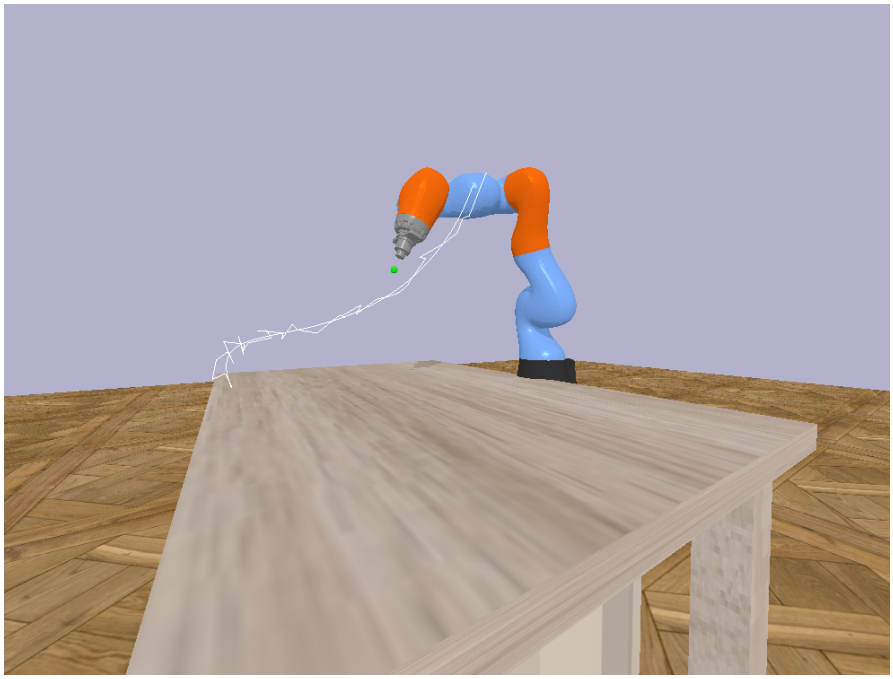
Po načtení filtru a trajektorie se zobrazí prostředí s trajektorií a je vizualizován proces filtrace způsobem popsaným v následujících kapitolách.

Zobrazení trajektorie

Trajektorie byla reprezentována diskrétní posloupností bodů, mezi kterými byla vytvořena úsečka metodou *pybullet.addUserDebugLine()*. Toto bylo provedeno pro čistá i zašuměná data. Ukázku můžeme vidět na obrázcích 3.3 a 3.4



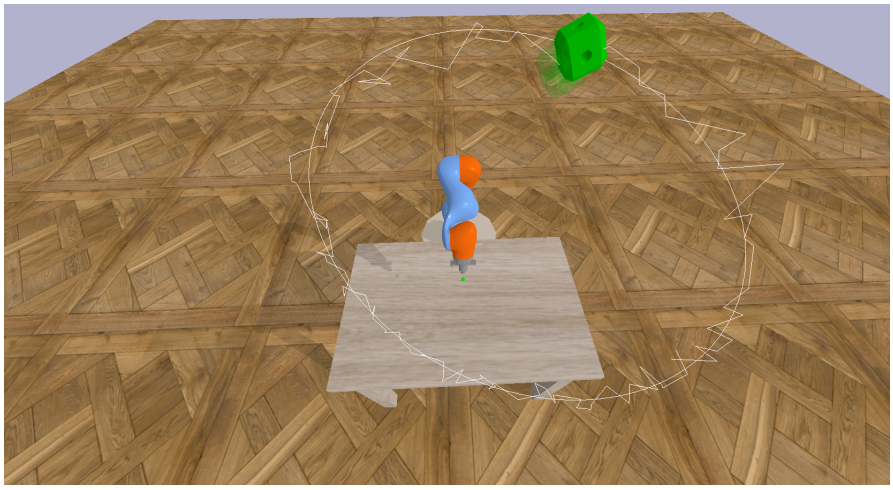
Obrázek 3.3: Ukázka vizualizace úsečky



Obrázek 3.4: Ukázka vizualizace splinu

- **Vizualizace pohybu a orientace předmětu po trajektorii**

Engine pybullet dovoluje vložit do scény objekty a nastovavat jejich pozici a orientaci. Pro vizualizaci "skutečného"pohybu byla použita kostka, které se posouvala po pozicích určených trajektorií a byla rotována podle rotační trajektorie. Pro vizualizaci zašuměných dat byla použita průhledná kostka. Ukázka na obr. 3.5.



Obrázek 3.5: Ukázka vizualizace pozice předmětu

- **Vizualizace částic filtru a odhadu pozice a orientace**

K vizualizaci částic filtru byly v prostředí vytvořeny malé "kuličky"pomocí metod *pybullet.createVisualShape()* a *pybullet.createMultiBody()*. K vizualizaci odhadu pozice a orientace byla stejným způsobem vytvořena tmavá větší kostka, která byla umisťována vždy do filtrem odhadované pozice předmětu, orientována dle výsledku filtru.

- **Animace procesu filtrace**

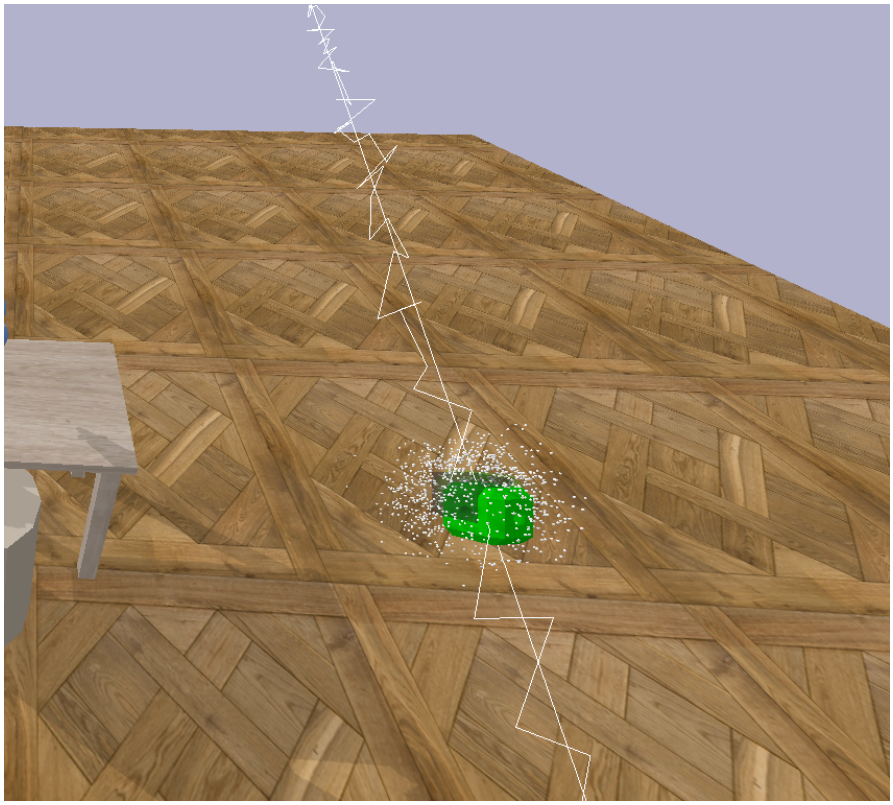
Celá animace procesu filtrace sestávala z přesouvání objektů do dalšího kroku podle algoritmu částicového filtru. Šlo tedy o přesouvání samotného předmětu do dalšího bodu na trajektorii, přesouvání částic filtru dle algoritmu a přesouvání kostky zobrazující odhad pozice a orientace. Vizualizovány byly pouze částice filtru sledující pozici objektu, částice pro odhad orientace objektu nebylo možné spolehlivě zobrazit. Mezi každým přesunutím proběhla pauza pro možnost analýzy situace.

Snímky z animace jsou na obr. 3.6

- **Společné rozhraní filtrů**

Jelikož je každý implementovaný filtr testován stejným skriptem, musí filtry obsahovat společné rozhraní. Pro každý filtr byla vytvořena třída jež obsahuje metody pro každý filtr společné. Tyto metody odpovídají jednotlivým krokům filtrace:

- Inicializace — ta probíhá přes metodu *apply_first_measurement*, neboť s žádnou počáteční informací o stavu objektu lze předpokládat, že se objekt vyskytuje v blízkosti prvního provedeného měření.



Obrázek 3.6: Ukázka vizualizace filtračního procesu v prostředí MyGym

- Predikce — pro každý filtr probíhá mírně odlišně.
- Aktualizace — stejně jako u predikce se liší pro každý filtr.
- Odhad stavu — uložení odhadu stavu objektu po provedení predikce a aktualizace.
- Převzorkování — platí pouze pro částicové filtry.
- Krok filtru — v případě, že průběh filtrace nechceme vizualizovat, je vhodné zavolat metodu *filter_step*, která provede celý krok filtru včetně predikce, aktualizace, odhadu stavu a v případě PF i převzorkování.

Následující kapitoly podrobněji popíší implementaci jednotlivých filtrů, zejména to, jak se liší od společného základu popsaného výše. Popis bude věnován filtrům pozičním — filtry orientace jsou tvořeny analogicky a jejich rozdíl bude popsán v jedné shrnující kapitole (kap. 3.8)

3.3 Kalmanův filtr

Ačkoliv výsledný algoritmus víceobjektového sledování bude díky svým vlastnostem obsahovat částicový filtr, pro srovnání výkonu samotné filtrace byl vytvořen také klasický Kalmanův filtr.

Jak již bylo zmíněno v kap. 3.1.1, pozice a rotace objektu lze považovat za vzájemně nezávislé. Filtrace pozice a orientace bude tedy probíhat odděleně. Protože je algoritmus filtrace pozice a orientace velmi podobný, budeme v následujících kapitolách pro jednoduchost uvažovat pouze 3D pozici objektu vyjádřenou vektorem $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}$. Úpravy filtrů potřebné k filtraci rotace jsou blíže popsány v kapitole kap. 3.8.

Implementace

Knihovna *filterpy* [11] poskytuje již hotovou implementaci Kalmanova filtru v jedné třídě a bylo pouze potřeba ji funkčně zakomponovat do výše uvedeného rozhraní.

V souboru *particle_filter.py* byla tedy vytvořena samostatná třída *myKalmanFilter*. Jejími hlavními funkcemi jsou

- Inicializace samotného Kalmanova filtru.
- Integrace do společného rozhraní.
- Zakomponování adaptivního filtrování.

Inicializace Kalmanova filtru

Pro správnou inicializaci Kalmanova filtru z knihovny *filterpy* je třeba určit

- Dimenzi stavu x a měření z.
- Hodnoty matic modelu $\mathbf{F}, \mathbf{H}, \mathbf{P}, \mathbf{R}$ a $\mathbf{Q}$.

Budeme pracovat s filtrem s konstantní rychlostí. Dimenze stavu bude tedy 6:

$$\tilde{\mathbf{x}} = \begin{bmatrix} x & \dot{x} & y & \dot{y} & z & \dot{z} \end{bmatrix} \tag{3.3}$$

kde $\tilde{\mathbf{x}}$ je vnitřní odhadovaný stav o objektu. Měření poskytuje údaj pouze o pozici objektu, jeho dimenze je tedy rovna 3.

Přechodový model filtru odpovídá klasickému modelu s konstantní rychlostí a protože měříme pozici objektu přímo, je i model pozorování jednoduchý:

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

Inicializace matice $\mathbf{P}$ není natolik podstatná, neboť ta se mění na základě získaných měření. Její hodnoty závisí na konkrétních parametrech trajektorie, rychlosti objektu atd. Více v kap. 5.1.

Matice Q by měla pro daný problém reprezentovat tzv. *diskrétní bílý šum*. Její hodnota by měla odpovídat střední hodnotě bílého šumu $\mathbf{w}$

$$\mathbf{Q} = \mathbb{E}[\mathbf{w}\mathbf{w}^T] \tag{3.4}$$

kde $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_p^2)$ — jde tedy o procesní šum.

Pro její výpočet knihovna *filterpy* disponuje funkcí *Q_discrete_white_noise*, jež přijímá parametry dimenze, dt a rozptyl. Protože model odpovídá modelu konstantní rychlosti, je pro každou ze souřadnic x, y, z dimenze $\mathbf{Q}$ rovna 2. Hodnoty dt a σ_p vyplývají z parametrů trajektorie a pohybu, viz. kap. 5.1.

Matice $\mathbf{R}$ má v našem případě rozměry 3×3 a inicializovány jsou pouze její hodnoty na diagonále o velikosti σ_z^2, což je hodnota rozptylu nepřesnosti v měření.

Integrace do společného rozhraní

Třída *myKalmanFilter* disponuje metodami predikce, aktualizace, *apply_first_measurement()*, odhadem stavu a krokem filtru pro integraci do společného rozhraní.

Metody predikce a aktualizace pouze použijí interní metody predikce a aktualizace u Kalmanova filtru, aplikace prvního měření pouze změní vnitřní stav filtru, odhad stavu pouze přečte odhadovaný stav z Kalmanova filtru a krok filtru zavolá po sobě funkce predikce, aktualizace a odhad stavu. Dále v sobě krok filtru obsahuje i **adaptivní filtraci** popsanou níže.

Adaptivní filtrace

Aby měl Kalmanův filtr vůbec šanci reagovat na měnící se trajektorii, byl do něj zakomponován jednoduchý mechanismus adaptivní filtrace [11]. Jeho princip spočívá v tom, že hodnota procesního šumu $\mathbf{Q}$ je proměnlivá a závisí na velikosti rezidua $\mathbf{y}_k$. Pokud je reziduum příliš velké (tedy měření se vzdaluje predikci), předpokládá filtr "manévr"objektu a výrazně zvýší hodnotu procesního šumu $\mathbf{Q}$. Tím se výrazně zvýší hodnota Kalmanova zisku $\mathbf{K}_k$ a výsledný odhad po kroku aktualizace se velmi přiblíží naměřené hodnotě. Filtr je tak rychleji schopen upravit svůj vnitřní stav dle měřených hodnot.

Toto se realizuje přes normalizovanou hodnotu čtverce rezidua

$$\varepsilon = \mathbf{y}^{\mathbf{T}}\mathbf{S}^{-\mathbf{1}}\mathbf{y} \tag{3.5}$$

kde $\mathbf{S}$ je kovarianční matice systémové nejistoty definovaná v rovn. 2.10. Pokud tedy ε přesáhne určitou konstantní hodnotu ε_{max} , vynásobí se $\mathbf{Q}$ faktorem Q_{scale}. Jakmile hodnota ε zase klesne, vrátí se $\mathbf{Q}$ na původní hodnotu. Konkrétní hodnoty ε_{max} a Q_{scale} uvedeny v kap. 5.1

3.4 Částicové filtry

Tato sekce se bude věnovat implementaci společných vlastností 3 vytvořených částicových filtrů.

Parametry

3 vytvořené částicové filtry mají několik odlišných parametrů. Avšak ty, které zůstaly všem společné, jsou:

- počet částic n,
- směrodatná odchylka procesního šumu σ_p,
- směrodatná odchylka šumu měření σ_z.

Důležitým parametrem je právě počet částic. Čím více částic filtr má, tím je přesnější a robustnější, tedy tím více je schopen se adaptovat na proměnlivost trajektorií. Na druhou stranu vyšší počet částic zvyšuje výpočetní náročnost a je tedy třeba filtr navrhnout tak, aby splňoval svůj úkol za použití spíše menšího počtu částic.

Metody

Důležitým úkolem při tvorbě PF je vytvoření metod pro jednotlivé kroky obecného algoritmu popsaného v kap. 2.1.2.

Inicializace částic

Inicializace částic byla provedena metodou *apply_first_measurement()*, jež využívá metodu *create_gaussian_particles()*. Částice se inicializují po prvním měření. Odhad o stavu objektu se nastaví na naměřenou hodnotu a poté se vytvoří částice. Každá ze souřadnic x, y, z pozice objektu každé částice je generována z Gaussovského rozdělení o střední hodnotě rovné odhadu stavu (naměřené hodnotě) se směrodatnou odchylkou rovné procesnímu šumu σ_p vynásobenou třemi (pro zvýšení počáteční nejistoty), tedy

$$\tilde{\mathbf{x}}_1^{(i)} \sim \mathcal{N}(\mathbf{z}_1, \mathbf{Q}) \tag{3.6}$$

kde $\tilde{\mathbf{x}}_1^{(i)}$ je vnitřní stav i-té částice filtru.

Predikce

Metoda predikce přesune každou částici na predikovanou pozici na základě vnitřního modelu pohybu filtru. Ty se liší dle jednotlivé implementace filtru, které budou blíže popsány v kapitolách 3.5, 3.7 a 3.6. Dále se k predikované pozici přidá v každé souřadnici Gaussovský šum se střední hodnotou 0 a směrodatnou odchylkou o hodnotě σ_p. Tento proces odpovídá lineárnímu stochastickému modelu pohybu z rovn. 2.1.

Aktualizace

Základ metody aktualizace funguje u všech tří filtrů v podstatě stejně a vychází z **Bayesova pravidla** v rovn. 2.7. Pro každou částici se vypočte její vzdálenost od naměřené hodnoty $\mathbf{z}_k$. Následně se vypočte váha částice dle toho, jak pravděpodobné (myšleno z angl. *likely*) je dané měření vzhledem k predikované pozici částice:

$$\tilde{w_k}^{(i)} \sim p(\mathbf{z}_k | \tilde{\mathbf{x}}_{k|k-1}^{(i)}) \tag{3.7}$$

kde $\tilde{w_k}^{(i)}$ je nenormalizovaná váha i-té částice a $\tilde{\mathbf{x}}_{k|k-1}^{(i)}$ je predikovaný stav částice i v kroku k. Váhy $\tilde{w_k}^{(i)}$ se dále normalizují, aby jejich součet byl roven jedné.

Víme, že pravděpodobnostní rozdělení chyby měření je rovné šumu měření. Je tedy Gaussovské s nulovou střední hodnotou a směrodatnou odchylkou rovné σ_z. Máme tedy:

$$p(||\mathbf{z}_k - \mathbf{x}_k||) \sim \mathcal{N}(0, \sigma_z^2) \tag{3.8}$$

Pokud vypočteme vzdálenost každé částice od měření:

$$p(d_k^{(i)}) \sim \mathcal{N}(0, \sigma_z^2) \tag{3.9}$$

kde $d_k^{(i)} = |\mathbf{z}_k - \tilde{\mathbf{x}}_{k|k-1}^{(i)}|$.

Protože pravděpodobnostní rozdělení chyby měření, tedy $p(\mathbf{z}_k | \tilde{\mathbf{x}}_{k|k-1}^{(i)})$ je rozdělená stejně jako $p(d_k^{(i)})$, jsou nenormalizované váhy částic $\tilde{w}_k^{(i)}$ vypočteny z rozdělení v rovn. 3.9 pro hodnoty $d_k^{(i)}$.

Odhad stavu

Základ odhadu stavu je opět pro všechny PF stejný. Spočítá se vážený průměr částic:

$$\hat{\mathbf{x}}_{\mathbf{k}|\mathbf{k}} = \sum_{i=1}^{n} w_k^{(i)} \tilde{\mathbf{x}}_{k|k-1}^{(i)} \tag{3.10}$$

Odhady stavů se ukládají do filtru pro následné zpracování.

Převzorkování

Převzorkování probíhá v případě, že na odhad stavu měl vliv pouze malý počet částic — tedy velké množství částic získalo v kroku aktualizace malou váhu. Počet částic, jejichž váhy nebyly příliš nízké a měly nezanedbatelný vliv na výsledný odhad stavu se spočítá pomocí tzv. **efektivního N** [11]:

$$N_{eff}^{(k)} = \frac{1}{\sum_{i=1}^{n} (w_k^{(i)})^2} \tag{3.11}$$

Pokud $N_{eff}^{(k)}$ nepřesáhne určitou hodnotu (typicky $\frac{n}{2}$, dojde k převzorkování. Metod převzorkování existuje vícero. Princip jejich fungování lze najít v [11], ve výsledku všechny metody odstraní částice s menšími vahami a nahradí je kopiemi částic s vahami většími.

Naše algoritmy budou používat metodu **stratified resampling** nebo **systematic resampling**, obojí je implementováno v knihovně *filterpy*.

Obnova částic

V případě, že objekt prudce mění trajektorii a částice filtru jsou moc daleko od měření (N_{eff} je moc nízké), dojde k nové inicializaci částic v místě měření. Tímto opatřením se vyhneme případné divergenci filtru.

3.5 Částicový filtr g-h

Nejdůležitější bod, ve kterém se jednotlivé částicové filtry liší, je predikce. Filtr totiž obsahuje vlastní model pohybu, na které predikci zakládá. částicový filtr g-h *(z angl. particle filter g-h)* (PFGH) implementuje model s globální rychlostí a akcelerací pro všechny částice. Rychlost i zrychlení se vypočítávají metodou tzv. g-h filtru, který bude níže popsán.

Parametry filtru

Kromě společných parametrů obsahuje PFGH ještě parametry:

- Směrodatná odchylka rychlosti σ_v
- Konstanta $g \epsilon [0; 1]$
- Konstanta $h \epsilon [0; 1]$

Obecný princip

V každé iteraci/kroku k má filtr k dispozici odhad globální rychlosti

$$\hat{\mathbf{v}}_k = \begin{bmatrix} \dot{x}_k & \dot{y}_k & \dot{z}_k \end{bmatrix} \tag{3.12}$$

a globální zrychlení

$$\hat{\mathbf{a}}_k = \begin{bmatrix} \ddot{x}_k & \ddot{y}_k & \ddot{z}_k \end{bmatrix} \tag{3.13}$$

.

Výpočet obou probíhá v kroku odhadu stavu, kdy se na základě odhadované změny pozice od předchozího kroku k vypočte odhad rychlosti. Nová globální rychlost se pak vypočítá váženým průměrem rychlosti z předchozího kroku a nové rychlosti, kdy parametr g určuje váhu rychlosti z předchozího kroku.

Stejný postup platí pro výpočet zrychlení, kdy parametr h určuje váhu zrychlení z předchozího kroku. Vypočtená hodnota zrychlení následně ještě změní hodnotu rychlosti.

V kroku predikce se následně posunou všechny částice dle globální rychlosti a přidá se ještě poziční šum se směrodatnou odchylkou σ_p.

Metody

Všechny metody kromě predikce a odhadu stavu se od standardního částicového filtru (kap. 3.4) neliší.

Predikce

Predikovaná pozice částic se vypočte následovně,

$$\tilde{\mathbf{x}}^{(i)}_{k|k-1} = \tilde{\mathbf{x}}^{(i)}_{k-1|k-1} + (\hat{\mathbf{v}}_{k-1} + \mathbf{u})dt + \mathbf{w} \tag{3.14}$$

kde $\mathbf{w} \sim \mathcal{N}(0, \sigma_p^2)$ a $\mathbf{u} \sim \mathcal{N}(0, \sigma_v^2)$

Odhad stavu

Odhad pozice se vypočte váženým průměrem. Dále se upraví hodnota rychlosti,

$$\bar{\mathbf{v}}_k = g \cdot \hat{\mathbf{v}}_{k-1} + (1-g)\frac{\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_{k-1}}{dt} \tag{3.15}$$

hodnota zrychlení,

$$\hat{\mathbf{a}}_k = h \cdot \hat{\mathbf{a}}_{k-1} + (1-h)\frac{\bar{\mathbf{v}}_k - \mathbf{v}_{k-1}}{dt} \tag{3.16}$$

a nakonec ještě jednou hodnota rychlosti

$$\hat{\mathbf{v}}_k = \bar{\mathbf{v}}_\mathbf{k} + \hat{\mathbf{a}}_k dt \tag{3.17}$$

kde $\bar{\mathbf{v}}_k$ je předběžná hodnota rychlosti před aplikováním zrychlení. Hodnota globální rychlosti i zrychlení je takto filtrována tzv. g-h filtrem.

3.6 Částicový filtr s Kalmanovým filtrem

částicový filtr s kalmanovým filtrem *(z angl. Kalman particle filter)* (KPF) opět počítá s globální rychlostí a akcelerací pro všechny částice. Rozdíl je v tom, že hodnoty globální rychlosti a akcelerace nejsou filtrovány g-h filtrem, nýbrž samostatným Kalmanovým filtrem.

Parametry

Oproti společným parametrům obsahuje KPF navíc:

- $\mathbf{Q}$ — kovarianční matice procesního šumu pro Kalmanův filtr
- $\mathbf{R}$ — kovarianční matice šumu měření pro Kalmanův filtr
- c_a — konstanta pro vliv zrychlení na procesní šum

Obecný princip

Opět, v každé iteraci/kroku k filtru jsou k dipozici odhady globální rychlosti a akcelerace $\hat{\mathbf{v}}_k$ a $\hat{\mathbf{a}}_k$. Obě dvě hodnoty jsou získávány Kalmanovým filtrem, který odhaduje globální rychlost na základě naměřených hodnot (vydělených dt pro získání rychlosti).

V kroku predikce je provedena i predikce Kalmanova filtru a v kroku odhadu stavu je uměle vytvořeno "měření"rychlosti na základě rozdílu odhadů v posledních pozicích, vydělených dt. Toto měření slouží jako vstup do Kalmanova filtru, který svým krokem aktualizace odhaduje rychlost.

Kalmanův filtr pro určení rychlosti

Úkolem vnitřního Kalmanova filtru pro KPF je zpřesnění odhadu globální rychlosti $\hat{\mathbf{v}}_k$. Filtr zároveň drží údaj i o zrychlení objektu. Vnitřní stav filtru tedy reprezentuje

$$\tilde{\mathbf{v}}_k = \begin{bmatrix} \dot{x}_k & \ddot{x}_k & \dot{y}_k & \ddot{y}_k & \dot{z}_k & \ddot{z}_k \end{bmatrix} \tag{3.18}$$

kde $\tilde{\mathbf{v}}_k$ je označení pro odhadovaný stav filtru v iteraci k. Odhad globální rychlosti $\hat{\mathbf{v}}_k$ pak dostaneme z prvních 3 prvků $\tilde{\mathbf{v}}_k$

Inicializace

Přechodový model (model pohybu) je stejný jako v případě samostatného Kalmanova filtru z kap. 3.3. Dostáváme tedy:

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

Inicializace hodnot $\mathbf{R}, \mathbf{P}$ a $\mathbf{Q}$ je také stejná jako u Kalmanova filtru. Samotné hodnoty však budou jiné, protože filtr počítá s rychlostí, která je nepřímo úměrná velikosti dt. Je také rozumé inicializovat počáteční odhad stavu $\tilde{\mathbf{v}}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ a počáteční hodnoty matice $\mathbf{P}$ velké, protože ze začátku nevíme, jakým směrem a jakou rychlostí se objekt pohybuje.

Metody

Všechny metody kromě predikce a odhadu stavu jsou totožné s ostatními částicovými filtry.

Predikce

Výpočet predikce se mírně liší od výpočtu predikce u PFGH:

$$\tilde{\mathbf{x}}^{(i)}_{k|k-1} = \tilde{\mathbf{x}}^{(i)}_{k-1|k-1} + \hat{\mathbf{v}}_{k-1} \cdot dt + ||\tilde{\mathbf{a}}_\mathbf{k}||\mathbf{w}_a + \mathbf{w}_p \tag{3.19}$$

kde $\tilde{\mathbf{a}}_k$ je odhad zrychlení z vnitřního Kalmanova filtru, $\mathbf{w}_a \sim \mathcal{N}(0, \sigma_a^2)$ je procesní šum škálovaný zrychlením a $\mathbf{w}_p \sim \mathcal{N}(0, \sigma_p^2)$ je konstantní procesní šum.

Tímto způsobem je do filtru zakomponován údaj z Kalmanova filtru o zrychlení objektu. Čím větší toto zrychlení je, tím větší je procesní šum, protože předpokládáme změnu trajektorie pohybu objektu.

Odhad stavu

Samotný odhad stavu je proveden standardním způsobem pomocí váženého průměru částic. Kromě toho se vygeneruje umělé měření rychlosti:

$$\mathbf{z}_v^{(k)} = \frac{\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_{k-1}}{dt} \tag{3.20}$$

Následně je zavolána funkce aktualizace vnitřního Kalmanova filtru pro toto vygenerované měření. Tím získáme odhad globální rychlosti $\hat{\mathbf{v}}_k$. Kalmanův filtr zároveň drží informaci o zrychlení objektu $\hat{\mathbf{a}}_k$.

3.7 Částicový filtr s rychlostí

Tento filtr se liší od předchozích v dimenzi svých částic. Každá částice v sobě nese i údaj o rychlosti objektu, čímž lze lépe skutečná rychlost objektu odhadnout. Filtr má potenciál být přesnější, ale kvůli vyšší dimenzionalitě částic může být výpočetně náročnější.

Obecný princip

Každá částice filtru obsahuje údaj o rychlosti objektu:

$$\tilde{\mathbf{x}}^{(i)}_k = \begin{bmatrix} x^{(i)}_k & y^{(i)}_k & z^{(i)}_k & \dot{x}^{(i)}_k & \dot{y}^{(i)}_k & \dot{z}^{(i)}_k \end{bmatrix} \tag{3.21}$$

Predikce filtru tedy posune částice dle své vlastní rychlosti, nikoli globální. Rychlost částice v každém kroku zůstává konstantní, pouze se k ní přidá šum. Tímto způsobem je možné modelovat akceleraci — ty částice, jejichž rychlost se vlivem šumu změnila dle trajektorie objektu, mají vyšší váhu a je nižší pravděpodobnost, že budou převzorkováním vyřazeny.

Dále bylo implementováno adaptivní filtrování, kdy jsou procesní šum σ_p a šum rychlosti σ_v proměnlivé na základě vypočteného rezidua (rozdílu mezi predikovanou a naměřenou pozicí).

- **Parametry**

  Kromě společných parametrů obsahuje filtr parametry:

  - Konstanta pro reziduál g_r
  - Konstanta pro šum rychlosti c_v
  - Konstanta pro procesní šum c_p

- **Metody**

Predikce

Pohyb částic v kroku predikce závisí na jejich vlastní rychlosti,

$$\tilde{\mathbf{x}}_k^{(i)}[:3] = \tilde{\mathbf{x}}_{k-1}^{(i)}[:3] + \tilde{\mathbf{x}}_{k-1}^{(i)}[3:] \cdot dt + \mathbf{w}_p \tag{3.22}$$

kde $\tilde{\mathbf{x}}_k^{(i)}[:3]$ označuje první 3 prvky stavového vektoru i-té částice (tedy pozici), $\tilde{\mathbf{x}}_{k-1}^{(i)}[3:]$ označuje poslední 3 prvky stavového vektoru i-té částice (tedy rychlost) a $\mathbf{w}_p \sim \mathcal{N}(0, \sigma_p)$.

Rychlost částic se změní o hodnotu šumu:

$$\tilde{\mathbf{x}}_k^{(i)}[3:] = \tilde{\mathbf{x}}_{k-1}^{(i)}[3:] + \mathbf{w}_v \tag{3.23}$$

kde $\mathbf{w}_v \sim \mathcal{N}(0, \sigma_v)$.

Hodnoty σ_v a σ_p jsou proměnlivé, závislé na konstantách c_p a c_v a jejich hodnota se určuje v kroku aktualizace.

Výpočet reziduálu

Před inicializací vah v kroku aktualizace se nejprve vypočte reziduál — tedy rozdíl mezi predikovanou pozicí a naměřenou pozicí:

$$\bar{\mathbf{y}}_k = \mathbf{z}_k - \hat{\mathbf{x}}_{\mathbf{k|k-1}}[:3] = \mathbf{z}_k - \frac{\sum_{i=1}^{n} \tilde{\mathbf{x}}_k^{(i)}[:3]}{n} \tag{3.24}$$

$\bar{\mathbf{y}}_k$ je předběžnou hodnotou reziduálu. Aby nedocházelo k přílišným výkyvům této hodnoty, je filtrována zakomponováním předchozích hodnot:

$$\mathbf{y}_k = g_r \cdot \mathbf{y}_{k-1} + (1 - g_r)\bar{\mathbf{y}}_k \tag{3.25}$$

Aktualizace

Inicializace vah probíhá standardním způsobem na základě vzdálenosti částic od naměřené hodnoty. Dále se v kroku aktualizace určí hodnoty σ_p a σ_v:

$$\sigma_p = c_p * ||\mathbf{y}_k|| \tag{3.26}$$

$$\sigma_v = c_v * ||\mathbf{y}_k||^2 \tag{3.27}$$

Tím je implementováno adaptivní filtrování. Pokud se hodnota reziduálu zvýší, pravděpodobně jde o změnu pohybu filtru, díky tomu se zvětší šum pohybu části a odhad stavu se více přiblíží hodnotě měření.

Všechny ostatní metody zůstávají od základní verze filtru nezměněné.

3.8 Specifika filtrů orientace

Hlavním důvodem, proč bylo potřeba vytvořit odlišné třídy pro filtry orientace je, že pro snadné provádění výpočtů je nejvhodnější reprezentovat 3D orientaci pomocí kvaternionu. Kvaternion obsahuje 4 prvky a proto nebylo možné použít předchozí filtry, které obsahovaly pouze 3 dimenze.

Filtry zpracovávají orientační kvaternion jako kdyby to byl standardní 4-dimenzionální vektor, tedy:

$$\hat{\mathbf{q}}_k = \begin{bmatrix} a_k & b_k & c_k & d_k \end{bmatrix} \tag{3.28}$$

Pro všechny filtry orientace byly upraveny rozměry všech stavových vektorů u Kalmanových filtrů a částic na 4 prvky.

Kvaternionová reprezentace navíc působila jeden problém, který bylo potřeba vyřešit. Pro jednu 3D orientaci totiž existují vždy 2 kvaterniony, které se liší pouze ve znaménku. Při změně rotační trajektorie se stávalo, že po provedené interpolaci další kvaternion "obrátil"znaménko, filtr tak zaznamenal významně odlišné hodnoty a trvalo nějakou dobu, než se na prudkou změnu adaptoval.

Řešení, které jsme navrhli a implementovali, řeší tento problém následovně:

Jakmile se nové měření přibližně rovnalo předchozímu odhadu stavu vynásobenému -1, okamžitě se odhad filtru také vynásobil -1, tedy pokud:

$$||\mathbf{z}_k + \mathbf{q}_{k-1}|| < c_f \tag{3.29}$$

kde c_f je konstanta, např. 0.1, pak změníme znaménko odhadu stavu:

$$\mathbf{q}_{k-1} \equiv -\mathbf{q}_{k-1} \tag{3.30}$$

a následně vypočteme standardním způsobem $\mathbf{q}_k$.

3.9 Způsob evaluace

Účinnost všech vytvořených filtrů bylo potřeba systematicky evaluovat. K tomu byla vytvořena sada dat v podobě trajektorií a následně byly filtry otestovány na těchto trajektoriích pro různé kombinace parametrů filtru.

Sada dat

V souboru *trajectory_generator.py* byly vytvořeny 3 třídy pro generátory trajektorií. Tyto generátory vyžadují zadání parametrů trajektorie uvedených v tab. 3.1. Trajektorie se generovaly následovně:

Úsečky a spline

(i) Určí se náhodný počet klíčových bodů z Poissonova rozdělení $n_{key} \sim Po(\lambda_p)$. Zajistí se, aby minimální počet bodů pro přímku byl 2 a pro spline 3.

(ii) Vygeneruje se n_{key} bodů v oblasti $\mathbf{B}_w$ (dolní a horní hranice pro souřadnice x, y, z). Zajistí se, aby minimální vzdálenost mezi body byla větší než d_{min}. Dále se vygeneruje

Název proměnné	Značení	Typ trajektorie
Směrodatná odchylka šumu měření	σ_z	Úsečky, kružnice, spline
Maximální úhlová rychlost	ω_m	Úsečky, kružnice, spline
Hranice pracovního prostoru	$\mathbf{B_w}$	Úsečky, kružnice, spline
Časový interval mezi měřeními	dt	Úsečky, kružnice, spline
Průměrná rychlost objektu	v	Úsečky, kružnice, spline
Střední hodnota klíčových bodů	λ_p	Úsečky, spline
Limit vzdálenosti bodů	d_{min}	Úsečky, spline
Minimální poloměr	r_{min}	Kružnice
Maximální poloměr	r_{max}	Kružnice

Tabulka 3.1: Tabulka parametrů pro generátory trajektorie

(iii) Vygeneruje se n náhodných orientačních kvaternionů (1 pro každý klíčový bod), mezi nimiž bude interpolována rotační trajektorie. Je zajištěno, aby mezi sousedními klíčovými body byl rozdíl v orientaci kvaternionů takový, aby nebyla překročena zvolená maximální úhlová rychlost ω. Tím se zajistilo, že generovaná trajektorie bude přirozená — tedy, že objekt nerotuje příliš rychle a tato úhlová rychlost se nemění mezi klíčovými body příliš prudce.

(iv) Klíčové body se proloží úsečkami či splinem. Dle rychlosti objektu (v případě splinu jde pouze o rychlost průměrnou, v jednotlivých částech trajektorie se mění) se trajektorie navzorkují. Vznikne tak n bodů v prostoru. Mezi orientačními kvaterniony v klíčových bodech se vytvoří interpolovaná rotace metodou *Quaternion.find_intermediates()*, která vytvoří n interpolovaných rotačních kvaternionů.

(v) Výsledné body na trajektorii se uloží do pole o rozměrech $n \times 3$ pro případ pozice a pro případ orientace o rozměrech $n \times 4$.

(vi) Tato poziční i orientační trajektorie představuje **ground truth** data. Následně se vytvoří jejich kopie, která je zatížena v každé souřadnici šumem se směrodatnou odchylkou σ_z.

POKUD ZBYDE ČAS, VYTVOŘIT OBRÁZEK

Kružnice

(i) Vygeneruje se náhodný poloměr r kružnice z intervalu $[r_{min}, r_{max}]$. Vygeneruje se náhodná normála kružnice $\mathbf{n} = \begin{bmatrix} n_x, n_y, n_z \end{bmatrix}$, kde $||\mathbf{n}|| = 1$

(ii) Vygeneruje se střed kružnice $\mathbf{x}_0 = \begin{bmatrix} x_0 & y_0 & z_0 \end{bmatrix}$ tak, aby celá kružnice byla v prostoru $\mathbf{B}_w$.

(iii) Z určených parametrů se vytvoří kružnicová trajektorie z následujících vzorců:

$$\varphi = \arctan 2(n_y, n_x) \tag{3.31}$$

$$\theta = \arctan 2(\sqrt{n_x^2 + n_y^2}, n_z) \tag{3.32}$$

$$x = x_0 - r \cdot (\cos t \sin \varphi + \sin t \cos \theta \cos \varphi) \tag{3.33}$$

$$y = y_0 + r \cdot (\cos t \cos \varphi - \sin t \cos \theta \sin \varphi) \tag{3.34}$$

$$z = z_0 + r \cdot (\sin t \sin \theta) \tag{3.35}$$

kde t je vektor lineárně interpolovaných hodnot $[0, 2\pi]$ o délce $n = \frac{2\pi r}{v}$.

(iv) Vygenerují se pouze 2 orientační kvaterniony za podmínek popsaných v algoritmu pro spline a úsečky a mezi nimi je interpolována rotační trajektorie.
(v) K poziční i rotační trajektorii je v každé souřadnici přidán šum se směrodatnou odchylkou σ_z

Způsob generování šumu

Vytvoření zašuměných dat o pozici objektu bylo jednoduché:

$$\mathbf{w}_x^{(k)} \sim \mathcal{N}(\mathbf{0}, \sigma_z) \tag{3.36}$$

$$\mathbf{z}_k = \mathbf{x}_k + \mathbf{w}_x^{(k)} \tag{3.37}$$

kde $\mathbf{w}_x^{(k)}$ je vektor šumu pro krok k.

Proces vytvoření šumu orientačního kvaternionu byl o něco složitější:

(i) Byl vygenerován náhodný kvaternion $\mathbf{q}_{axis}$, který určoval osu rotace kvaternionu šumu $\mathbf{q}_{noise}$.
(ii) Určila se náhodná hodnota úhlu rotace (α_q) kvaternionu šumu se směrodatnou odchylkou $\sigma_q * \sqrt{3}$ [1]
(iii) Vytvořil se kvaternion $\mathbf{q}_{noise}$ pomocí osy kvaternionu $\mathbf{q}_{axis}$ a úhlu α_q. Tento kvaternion se uložil do seznamu naměřených hodnot orientace.

Metoda evaluace

Filtry byly evaluovány na vygenerovaných trajektoriích dle dvou kritérií:

- průměrná čtvercová chyba *(z angl. mean-squared-error)* (MSE)
- Doba trvání filtrace

K evaluaci a nalezení optimálních parametrů byla využita knihovna *scikit-learn*[14] — konkrétně třída *RandomizedSearchCV()*, která primárně slouží k testování hyperparametrů u strojového učení, lze ji však využít i v našem případě.

[1] Tato odchylka zadána ve stupních (°) byla vynásobena faktorem $\sqrt{3}$. Je to proto, že směrodatná odchylka (SO) σ_q by měla odpovídat stupňové odchylce šumu v jedné dimenzi. Protože však máme dimenze 3, je nutné tuto odchylku zvětšit, protože $\sigma = \sqrt{\sigma_q^2 + \sigma_q^2 + \sigma_q^2} = \sqrt{3}\sigma_q$.

4 Implementace víceobjektového sledování

Pro implementaci víceobjektového sledování bylo možné navázat na implementaci samotné filtrace, a to jak v oblasti vizualizace a generování dat, tak v oblasti implementace algoritmu. Bylo však potřeba rozšířit funkce evaluačního skriptu, funkce skriptu generování a zpracovávání (konverze) dat (trajektorií), aby mohl být algoritmus víceobjektového sledování řádně otestován.

4.1 Parametry zadání

Víceobjektové testování má být testováno na scénáři obsahující různé množství trajektorií (odhadem 5–10). Typy trajektorií zůstávají stejné jako pro testování samotné filtrace (viz kap. 3.1.1). Parametry scénáře, na kterém budeme algoritmus testovat, jsou:

- Skutečné hodnoty trajektorie objektu (*ground truth*) i zašuměná měření jsou vzorkována v **nepravidelných** časových intervalech
- Počet trajektorií (a tedy i množství objektů) je algoritmu znám, včetně jejich odhadovaných tříd (třídou objektu se rozumí, zda jde o např. krychli, sféru nebo třeba hrnek)
- Objekty se mohou v určitou chvíli vyskytovat mimo scénu, ale mohou se na ní opět vrátit — algoritmus by měl umět s touto možností počítat
- Není zaručeno, že v každém kroku filtru dostaneme měření od všech objektů — některé objekty mohou být **okludovány** nebo zkrátka nezachyceny detektorem (například vlivem špatných světelných podmínek)
- Data mohou pocházet z více kamer, a tak se naopak v každém kroku může vyskytnout více měření jednoho objektu
- Data o měření z detektoru mohou být velmi nepřesná — občas se na scéně objeví měření, které zřejmě nepřísluší žádnému objektu

4.2 Generování dat

Formát dat

Pro účely víceobjektového sledování bylo potřeba změnit formát dat, který je použitý pro účely samotné filtrace. Zásadními rozdíly oproti formátu použitým ve filtraci jsou:

- Počet měření v každém kroku může být vyšší než jedna — je tedy třeba pro určení daného měření v kroku k přidat index j
- Zahrnutí času měření $t_{k,j}$
- Zahrnutí třídy měření $\gamma_{k,j}$

Z toho vyplývají úpravy datového formátu, který bude vstupovat do procesu filtrace a sledování – data o měření v sobě budou obsahovat údaj o času měření $t_{k,j}$ a odhadované třídě objektu $\gamma_{k,j}$:

$$\mathbf{z}_{k,j} = \begin{bmatrix} x_{k,j} & y_{k,j} & z_{k,j} & t_{k,j} & \gamma_{k,j} \end{bmatrix} \tag{4.1}$$

Data o orientaci objektu se rozšíří ve stejném formátu. V každém čase měření pozice dostaneme i měření orientace, které samozřejmě bude mít stejnou odhadovanou třídu jako příslušné měření pozice:

$$\mathbf{z}_{k,j}^{(\mathbf{q})} = \begin{bmatrix} a_{k,j} & b_{k,j} & c_{k,j} & d_{k,j} & t_{k,j} & \gamma_{k,j} \end{bmatrix} \tag{4.2}$$

Komplikovanější způsob evaluace formát dat dále zesložiťuje díky následujícím faktorům:

- Algoritmus musí vytvořit odhady pozice objektů i ve chvílích, kdy nevzniklo žádné měření
- I když intervaly mezi jednotlivými měřeními jsou nepravidelné, algoritmus stále musí periodicky fungovat a zohledňovat několik měření získaných v různých časech během jednoho kroku — je třeba tedy do dat o trajektorii obsáhnout i hodnoty nejen v okamžiku měření, ale i v periodických intervalech evaluace kroku algoritmu
- Je potřeba implementovat způsob vyhodnocení asociace dat
- Je potřeba vytvořit vhodný formát pro systematickou evaluaci pomocí *RandomizedSearchCV()*

Všechny tyto faktory budou zahrnuty do formátu vygenerovaných dat. Výsledkem generování dat budou celkem čtyři pole, a to $\mathbf{Y}, \mathbf{X}, \mathbf{Y}_{rot}, \mathbf{X}_{rot}$, jejichž formát je popsaný v následujících podkapitolách 4.2.1.

Skutečná data (*ground truth*)

Skutečná data jsou při vyhodnocení pomocí *RandomizedSearchCV()* označena vektorem (i vícedimenzionálním) $\mathbf{Y}$. Protože generujeme data poziční i orientační, rozdělíme jejich označení na $\mathbf{Y}$ a $\mathbf{Y}_{rot}$ Pole $\mathbf{Y}$ tedy bude složené z řádků se skutečnými hodnotami pozic objektů.

Teoreticky by stačilo mít skutečná data pouze v pravidelných časových intervalech. Metoda *RandomizedSearchCV* však spolehlivě přijímá data se stejnou dimenzionalitou $\mathbf{X}$ a $\mathbf{Y}$ a vyplatí se je tak vytvářet.

Data o pozicích objektu (pro jednu trajektorii) byla generována standardním způsobem dle zvolené trajektorie (více viz. kapitoly 3.9.1,) Během pozic byl zároveň vytvořen časový vektor $\mathbf{t}$ — K tomuto účelu byl určen parametr σ_t, tedy směrodatná odchylka časového intervalu. Byl vygenerován základový časový vektor

$$\mathbf{t}_b = \begin{bmatrix} 0 & 0,2 & 0,4 & \dots & t_{max} \end{bmatrix} \tag{4.3}$$

dle počtu bodů trajektorie. K tomu byl přidán časový vektor šumu $\mathbf{t}_{noise}$, který byl vygenerován pomocí n vzorků z Gaussovského rozdělení $\mathcal{N}(0, \sigma_t^2)$, kde n je délka vektoru $\mathbf{t}_b$.

Takto se vytvořil zadaný počet náhodných trajektorií (N) spolu s časovými vektory. Ke každé trajektorii byla přidána třída objektu dle uživatelského vstupu. Vzniklo tak N polí $\mathbf{Y}$ s prvky:

$$\mathbf{Y} = \begin{bmatrix} x_{1,i} & y_{1,i} & z_{1,i} & t_{1,i} & \gamma_i \\ x_{2,i} & y_{2,i} & z_{2,i} & t_{2,i} & \gamma_i \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n,i} & y_{n,i} & z_{n,i} & t_{n,i} & \gamma_i \end{bmatrix} \tag{4.4}$$

kde $i = 1, \dots, n$ označuje index objektu. Tyto pole poskládáme pod sebe dohromady v jedno.

Nyní je ještě potřeba doplnit data o o trajektorii v pravidelných intervalech po $0,2s$, které slouží pro určování okamžiku periodických operací algoritmu (predikce a aktualizace).

To bylo provedeno (pro všechny trajektorie) analogicky a následně bylo pole seřazeno podle času.

Skutečná data o orientaci objektu byla získána souběžně — jednotlivé orientace mezi klíčovými body byly interpolovány pomocí metody *Quaternion.slerp()*. Výsledkem bylo seřazené pole dle časů:

$$\mathbf{Y}_{rot} = \begin{bmatrix} a_i & b_i & c_i & d_i & t_i & \gamma_i \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_i & b_i & c_i & d_i & t_i & \gamma_i \end{bmatrix} \tag{4.5}$$

Protože nelze určit, která měření (z které z N) trajektorií bude v jakém pořadí, nejsou u měření uvedeny indexy k.

Zašuměná data

Zašuměná data jsou při vyhodnocení pomocí *RandomizedSearchCV()* označena vektorem (i vícedimenzionálním) $\mathbf{X}$. Podobně jako u dat skutečných, protože generujeme data poziční i orientační, rozdělíme jejich označení na $\mathbf{X}$ a $\mathbf{X}_{rot}$

Pro časové okamžiky samotných měření (nepravidelné intervaly) byla data po vytvoření skutečných dat dotvořena standardním způsobem, stejně jako v případě jedné trajektorie (popsáno v 3.9.1). Nyní bylo potřeba dotvořit zašuměná data pro časové násobky $dt = 0,2s$, ve kterých však ve skutečnosti žádná měření nemáme. Tyto okamžiky označíme hodnotou pozice či orientace, kterou budeme onu skutečnost absence měření rozpoznávat. Hodnota může být libovolná (například *numpy.NaN*). V našem případě byla zvolena hodnota, která se nachází daleko mimo dosah možných měřených hodnot (konkrétně vektor $\begin{bmatrix} 88 & 88 & 88 \end{bmatrix}$ u pozice a vektor $\begin{bmatrix} 88 & 88 & 88 & 88 \end{bmatrix}$ u orientace.

Systém generování

Data v požadovaném formátu byla generována pomocí třídy *MultipleTrajectoryGenerator*. Inicializace objektu této třídy vyžaduje tři parametry:

(i) Počet trajektorií — určení počtu trajektorií ve vygenerovaných scénářích
(ii) Parametry trajektorií — slovník, který obsahuje parametry a hodnoty, které se mají použít pro jednotlivé generátory trajektorií (sérií úseček, kružnic a splinů)
(iii) Standardní odchylku šumu časového intervalu — byla použita ke generování měření nikoliv v pravidelných intervalech jako v případě samotné filtrace, nýbrž v intervalech nepravidelných, které se odchylovaly od pravidelných hodnot na základě míry tohoto šumu

Dále třída obsahuje metody:

- *generate_1_scenario()* — hlavní metoda, která za použití jiných metod třídy kompletně vytvoří scénář víceobjektového sledování. Jejím výstupem jsou 4 dvoudimenzionální pole *X, y, X_rot, y_rot* ve formátu popsaným v kap. 4.2.1.
- *initialize_generator()* — metoda, která inicializuje náhodný generátor dle parametrů nastavených při inicializaci objektu této třídy
- *generate_trajectories()* — metoda, která postupně vygeneruje trajektorie z jednotlivých generátorů
- *initialize_X_and_y()* — metoda, která inicializuje pole $\mathbf{X}, \mathbf{Y}, \mathbf{X}_{rot}$ a $\mathbf{Y}_{rot}$ dle požadovaných dimenzí

- *save_1_trajectory()* — metoda, která vygenerovaný scénář v podobě polí $\mathbf{X}, \mathbf{Y}, \mathbf{X}_{rot}$ a $\mathbf{Y}_{rot}$ uloží do složky *./dataset/MOT/*

4.3 Vizualizace

Po vygenerování dat ve správném formátu pro evaluaci bylo nutné trajektorie vizualizovat. To bylo opět provedeno ve skriptu *filter_test.py*.

Přitom bylo žádoucí zachovat funkci vizualizačího skriptu i pro filtraci s pouze jedním objektem, a tak byl rozšířen systém zadávání uživatelského vstupu.

Rozšíření zpracování uživatelského vstupu

Skript *filter_test.py* byl celkově přepracován pro zahrnutí:

- generování trajektorií nebo scénářů pro uložení do sady dat,
- vizualizace procesu filtrace nebo víceobjektového sledování,
- vygenerování trajektorie nebo scénáře pro okamžitou vizualizaci filtrace.

Princip funkce skriptu *filter_test.py* je zobrazen na diagramu 4.1. Přesný diagram popisující sekvenci zpracování uživatelského vstupu pro skript *filter_test.py* je zobrazen v příloze na diagramu A.1.

Vizualizace více trajektorií

Vstupní data do vizualizačního skriptu ve formátu $\mathbf{X}, \mathbf{Y}$ byly nejprve převedeny do seznamu trajektorií ve stejném formátu, jaký byl použit pro filtraci 1 objektu (viz kap. 3.2.3). Poté byly všechny trajektorie vykresleny ve vizuálním prostředí stejným způsobem, jako byly vykresleny trajektorie u filtrace.

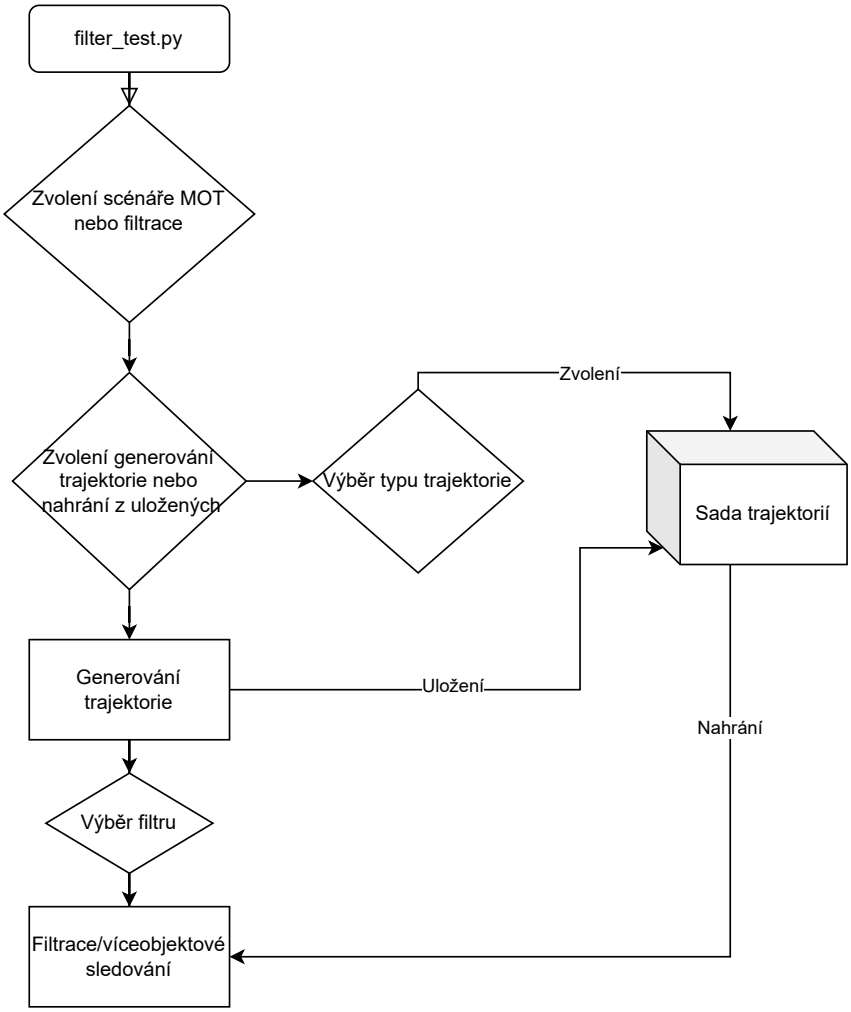
V případě, že uživatel zadal možnost generování trajektorií (ať už pro jejich uložení nebo okamžitou vizualizace filtrace/sledování), bylo potřeba zajistit, aby bylo možné zobrazenou trajektorii smazat ze scény a uvolnit tak místo pro trajektorii novou. To se provedlo pomocí metody *pybullet.removeAllUserDebugItems()*.

Narozdíl od vizualizování procesu filtrace byla přidána vizualizace úseků měření trajektorie – tím bylo možné zobrazit nepravidelnost časových intervalů nebo rozdíl mezi úsečkou s konstantní rychlostí a úsečkou s akcelerací. Jednotlivé body trajektorie, kde došlo k měření, byly vyznačeny malou černou kostkou. I tyto kostky bylo potřeba ze scény smazat a tak jejich identifikační číslo vygenerované při jejich přidání do scény bylo uloženo do pole, díky kterému bylo možné kostky smazat metodou *pybullet.removeUserDebugItem()*

Vizualizace odhadů, měření a částic

Vzhledem k většímu množství trajektorií bylo do scény přidáno více kostek zobrazující odhady a měření v jednotlivých časových okamžicích. Měření byly zobrazovány průhlednějšími žlutými kostkami a skutečné hodnoty sytějšími zelenými kostkami.

Vzhledem k dlouhému trvání vizualizace jednotlivých částic (hlavně při jejich inicializaci — přidání do scény) byla přidána možnost uživatelského vstupu zvolit si, kolik částic filtru se bude na scéně zobrazovat. Uživatel může zvolit buď zobrazení všech částic, pouze několika (v základu 40), nebo vůbec žádných.



Obrázek 4.1: Diagram fungování vizualizačního skriptu

Animace víceobjektového sledování

Animace víceobjektového sledování probíhala pomocí stejné sekvence jako při filtraci. Jelikož jsou hodnoty ve vstupních datech $\mathbf{X}$ seřazeny podle času, postupně se provádí kroky predikce, aktualizace, odhad stavu a převzorkování pro jednotlivá chronologicky přicházející měření.

4.4 Algoritmus víceobjektového sledování

V této sekci se zaměříme samotné implementaci algoritmu víceobjektového sledování dle metod popsaných v 2.2.

Návrh algoritmu

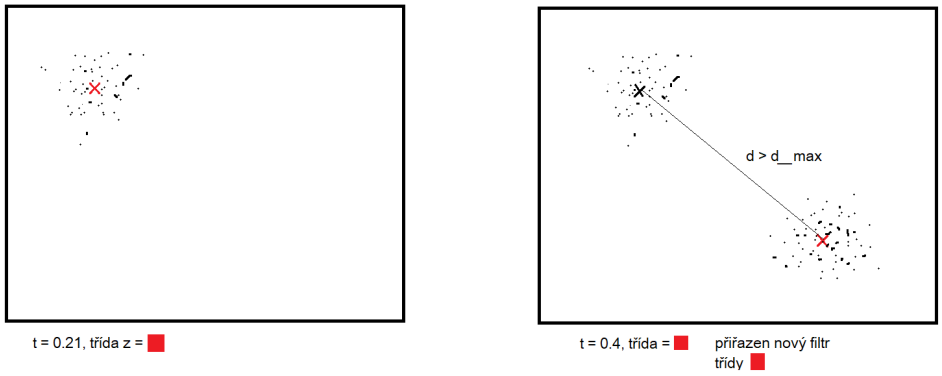
Důležitou vlastností řešené úlohy je, že je předem znám maximální počet objektů na scéně a jejich třídy. To nám umožňuje delegovat každému objektu jeden samostatný filtr a tím i sadu částic.

Protože však z počátku nevíme, kde se objekty na scéně mohou objevit, je potřeba inicializovat filtr až po vyhodnocení prvních měření.

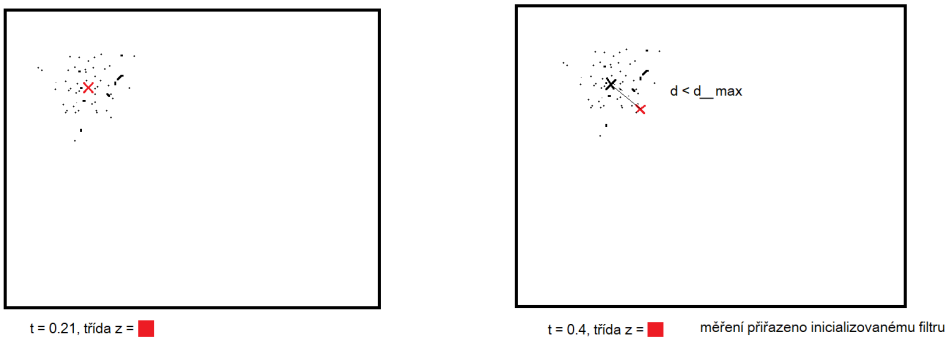
Inicializace filtrů

Princip inicializace filtrů spočívá v postupném inicializování filtrů k třídě měření, které se objeví na scéně, dokud není ke každému objektu z každé třídy (jejichž počet je znám) asociován filtr. Při obdržení měření třídy, u které ještě nebyl inicializován filtr, dojde automaticky k inicializaci filtru k tomuto měření (metodou *apply_first_measurement()*, viz kap. 3.4.2).

Je potřeba vyřešit situaci, kdy obdržíme další měření třídy, u které již byl nějaký filtr inicializován, ale ještě aspoň jeden filtr třídy iniciovat zbývá. Objekt může příslušet buď již inicializovanému filtru, nebo může jít o nový objekt. Tento problém lze vyřešit stanovením hranice vzdálenosti, za kterou je již měření považováno za příslušící jinému objektu. Situace je znázorněna na obrázcích 4.2a a 4.2b.



(a) Inicializace nového filtru vzdálenému měření



(b) Aktualizace inicializovaného filtru dle blízkého měření

Obrázek 4.2: *Proces inicializace filtrů stejné třídy – buď je inicializován nový filtr, když je nové měření dostatečně daleko, nebo je měření přiřazeno již inicializovanému filtru, pokud je měření blízko jeho odhadu*

Je třeba vhodně zvolit hodnotu d_{max}

Algoritmus

Jakmile byly všechny filtry inicializovány, nebo pokud obdržíme měření přiřazené již ke stávajícímu filtru, provádí se kroky algoritmu.

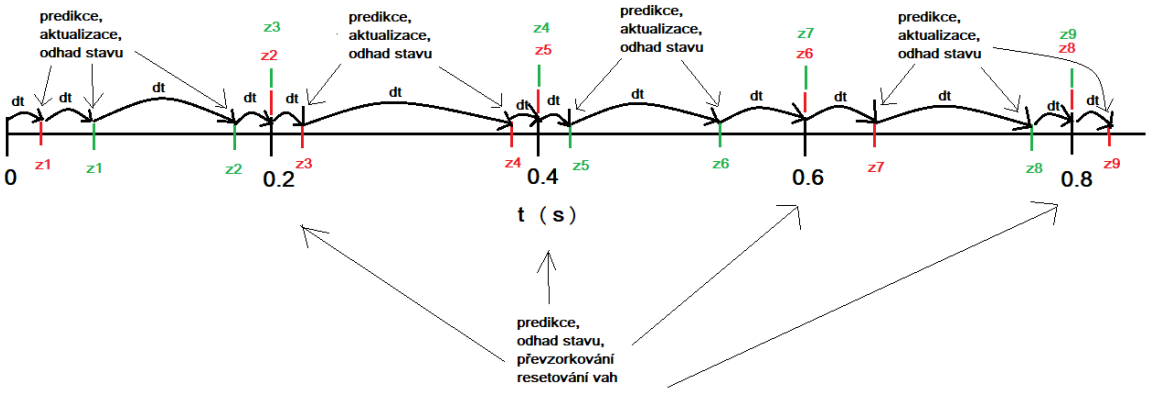
Důležitým aspektem při tvorbě algoritmu je fakt, že měřená data přicházejí v nepravidelných časových intervalech. Pokud chceme aktualizovat váhy částic filtru dle příchozího měření, musíme prvně provést predikci filtru – ta však musí počítat s časovou změnou odpovídající době uplynulé od předchozího měření. Kroky algoritmu lze rozdělit dle dvou situací:

(i) Příchozí měření přišlo v **nepravidelném** časovém intervalu a **obsahuje** naměřenou hodnotu. V tomto kroku se provede predikce všech filtrů a dle metod algoritmu PDA aktualizují váhy jednotlivých filtrů na základě přijatého měření.

(ii) Příchozí měření přišlo v **pravidelném** časovém intervalu (násobek $dt = 0,2s$) a **neobsahuje** naměřenou hodnotu. V tomto kroku se provede predikce všech filtrů a na základě naakumulovaných vah částic se provede odhad stavu filtru, který bude poté porovnáván se skutečnou hodnotou.

V zásadě má tedy algoritmus stále periodickou povahu a celý "krok" filtru se děje každých $dt = 0,2s$. Během této doby se akumulují v částicích filtrů váhy na základě aktualizací z měřených hodnot.

Proces algoritmu, respektive posloupnost jeho jednotlivých kroků je znázorněn na obr. 4.3:



Obrázek 4.3: Ukázka časové osy víceobjektového sledování pro dva objekty rozdílných tříd

Na obrázku 4.3 vidíme časovou osu, na které jsou zobrazena jednotlivá měření označená jako $\mathbf{z}_i$, kde jejich barva určuje třídu detekovaného objektu. Okamžiky v násobcích času $0,2s$ jsou zvýrazněny, neboť zde se vykonává kompletní krok algoritmu. V těchto okamžicích přichází algoritmu fiktivní hodnota měření od každého objektu, která pouze dává algoritmu informaci, že má pro daný filtr vytvořit odhad stavu, převzorkovat částice a resetovat jejich váhy. Tím se naakumulované váhy částic ze všech měření během intervalu $0,2s$ projeví do odhadu stavu v tomto okamžiku, dále se projeví i v částicích, které zůstanou přítomny po převzorkování a nakonec se obnoví váhy pro novou iteraci algoritmu. Po těchto krocích se zároveň zvýší hodnota k označující krok sledovacího algoritmu.

V okamžicích měření, která se nacházejí mimo násobky $0,2s$, dojde k predikci částic každého filtru (dle doby dt, která uplynula od posledního zpracovaného měření), a následné aktualizaci vah částic každého filtru na základě měření. Jednotlivé metody algoritmu (predikce, aktualizace, odhad stavu) jsou popsány v následujících kapitolách.

- **Metody algoritmu**

Predikce

Metoda predikce se v zásadě podstatně od predikce u filtrace neliší. Jediný rozdíl je v nutnosti správně měřit uplynulý čas od posledního měření. Predikce se provede pro každý filtr zvlášť, neboť měření jakékoliv třídy má vliv na každý filtr. Tento princip vychází z algoritmu PDA popsaného v kapitole 2.2.3.

Aktualizace

V metodě aktualizace se nachází jádro použitého algoritmu. Zde se uplatňuje princip PDA nebo JPDA a implicitně se provádí asociace dat dle asociačních hypotéz.

Fungování kroku aktualizace v implementovaném algoritmu lze popsat následovně:

- Při inicializaci filtru, nebo pokud zrovna byly resetovány váhy, jsou váhy $\tilde{w}_{k,i}^{(l)}$ nastaveny na hodnotu **nulové asociační hypotézy** (více viz kap. 4.4.2). To je provedeno pro zajištění akumulace vah.
- Pro každou částici každého filtru a dané měření $\mathbf{z}_{k,j}$ se vypočte váha $w_{k,i,j}^{(l)}$ kde i značí index filtru a $l = 1, 2, \ldots, N$ značí index částice filtru. Tato váha je vypočtena jako funkční hodnota Gaussovského rozdělení $\mathcal{N}(\mathbf{z}_{k,j}, \mathbf{R})$ kde $\mathbf{R}$ je kovarianční matice šumu měření s hodnotami σ_z^2 na diagonále. Tento výpočet váhy je stejný jako v kroku aktualizace v případě filtrace.
- Na základě naměřené třídy $\gamma_{k,j}$ je tato váha vynásobena koeficientem k_{cl}, který snižuje danou váhu pro měření jiné třídy, než ke které je filtr asociován.
- $\tilde{w}_{k,i,j^{(l)}}$ vynásobená k_{cl} se nyní přičte k dosavadní váze částice.

Koeficient třídy k_{cl} se vypočte z *matice tříd* $\mathbf{C}$:

$$k_{cl} = \mathbf{C} \cdot \boldsymbol{\gamma} \qquad \mathbf{C} = \begin{bmatrix} 1 & c_{1,2} & \ldots & c_{1,n_{cl}} \\ c_{2,1} & 1 & \ldots & c_{2,n_{cl}} \\ \vdots & \ddots & \ddots & \vdots \\ c_{n_{cl},1} & \ldots & \ldots & c_{n_{cl},n_{cl}} \end{bmatrix} \tag{4.6}$$

kde $\boldsymbol{\gamma}$ je vektor, který obsahuje hodnotu 1 na pozici $\gamma_{k,j}$ a prvky $c_{i,j}$ určují míru podobnosti mezi objekty i a j a tím jakousi pravděpodobnost, že byla špatně detekována třída objektu.

Tyto hodnoty lze libovolně nastavit (včetně hodnot 1 na diagonále!) dle vlastností detektoru a detekovaných objektů. Pro třídy objektů i a j, které jsou si podobné, a detektor mezi nimi častěji zaměňuje, je vhodné koeficient $c_{i,j}$ zvýšit. To lze provést například na základě empirických dat o záměně tříd detektoru.

Odhad stavu

Odhad stavu se pro každý filtr vypočte vždy po každém měření včetně pravidelných intervalů po $0,2s$. Výpočet je proveden standardním způsobem a to váženým průměrem pozic

(a orientací) všech částic (viz kap. 3.4.2). V tomto kroku algoritmu se projeví dosud naakumulované váhy částic, které byly měřením v daném kroku k nejblíže. Větší množství částic zajišťuje implicitní výpočet marginálních asociačních pravděpodobností — čím více částic a jejich vah je ovlivněno některým měřením, tím pravděpodobnější je asociace tohoto měření k objektu (tedy $\beta_{i,j}$).

Teoreticky je možné, že během kroku k nebylo zaznamenáno na scéně žádné měření $\mathbf{z}_{k,j}$. V takovém případě zůstaly váhy všech částic $\tilde{w}_{k,i}^{(l)}$ rovny *nulové asociační hypotéze* a jsou tedy stejné — provede se tedy rovnoměrný vážený průměr částic.

Převzorkování

Krok převzorkování opět proběhne standardním způsobem popsaným v kap. 3.4.2. Krok převzorkování je prováděn v pravidelných časových intervalech po $0,2s$ a pouze pokud velikost $N_{eff} < \frac{n}{2}$.

Obnova vah

Po každém kroku, kdy byl proveden odhad stavu a převzorkování, je opět potřeba obnovit váhy částic. Ty se narozdíl od problému samotné filtrace inicializují na hodnotu závislou nikoliv pouze na počtu částic $\frac{1}{N_i}$, nýbrž dle nulové asociační hypotézy.

Nulová asociační hypotéza je hypotéza, že se objektu nepřiřadí žádné z měření z kroku k. Tuto možnost chceme jistě zahrnout. Pokud nepřiřadíme objektu žádné měření, nelze logicky při výpočtu odhadu stavu váženým průměrem zvýšit váhu některých částic oproti ostatním a je tedy třeba přiřadit částicím váhu rovnoměrně. Jak velká má však váha příslušící nulové asociační hypotéze být, závisí na pravděpodobnosti absence měření v daném kroku.

Pravděpodobnost absence měření je dána průměrným počtem měření příslušícím daném objektu za jeden krok filtru. Pro příklad uveďme průměrný počet měření příslušícím danému objektu za jeden krok filtru $0,9$. To znamená, že v 10% trajektorie je objekt okludován, nebo ho zkrátka detektor nezachytí.

I při průměrném počtu 1 měření na jeden krok filtru však dosáhneme přibližně 25 % pravděpodobnosti, že v daném kroku filtr nebude obsahovat žádné měření. To lze pozorovat v příkladovém obrázku 4.3 pro červený objekt mezi časy $0,4s$ a $0,6s$ nebo pro zelený objekt mezi časy $0,2s$ a $0,4s$.

Pro získání hrubého odhadu pravděpodobnosti nulové asociační hypotézy spočteme podíl $\frac{0,25}{0,9} \doteq 0,2778$. Tato hrubá pravděpodobnost odpovídá hodnotě Gaussovského rozdělení $\mathcal{N}(0|\sigma_z^2)$ vzdálenosti částice od měření:

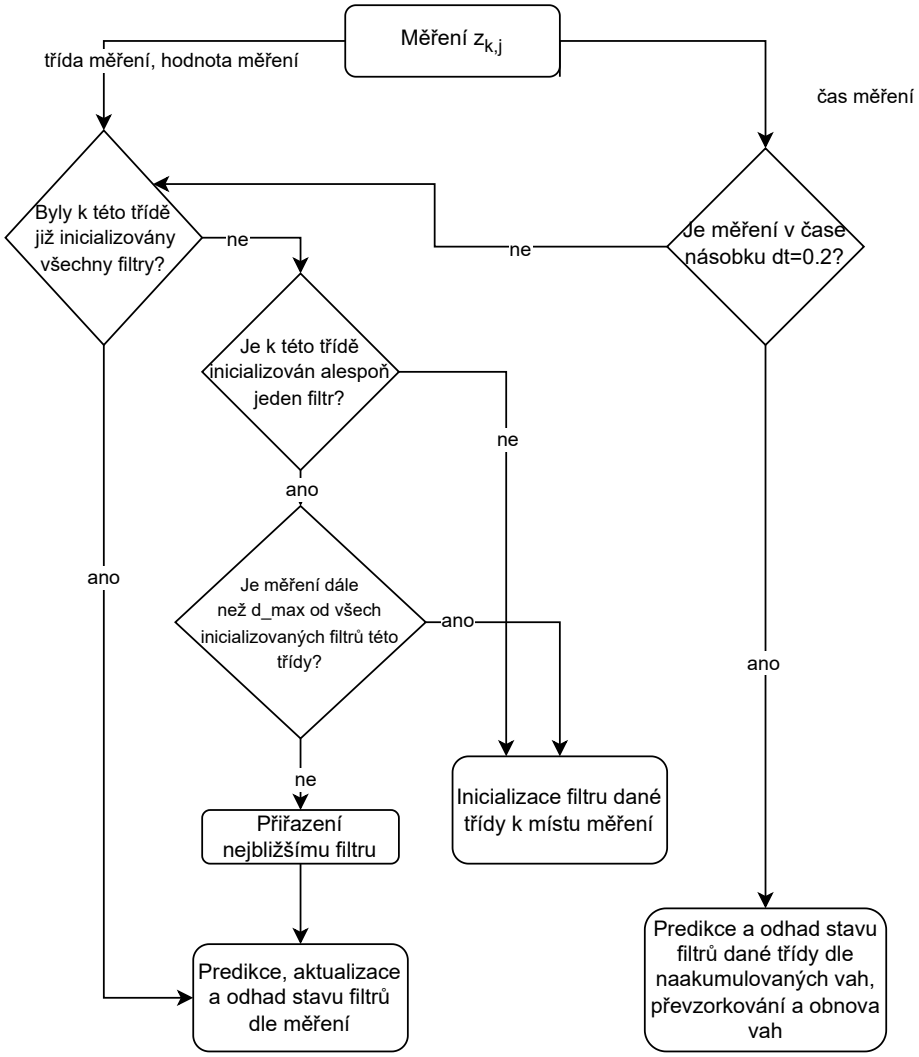
$$d = ||\mathbf{z}_{k,j} - \tilde{\mathbf{x}}_{k,i}^{(l)}|| \tag{4.7}$$

pro hodotu $d = 1,0853 \cdot \sigma_z$. Měření, které jsou částici blíže než tato vzdálenost (a zároveň jsou odpovídající třídy $\gamma_{k,j}$, přispějí váhou větší, než je váha nulové asociační hypotézy.

Obnova vah nastaví tedy prvotní ($w_{k,i,0}$ hodnotě Gaussovského rozdělení ve vzdálenosti $d = \sigma_z \cdot 1,0853$.

- **Procesní diagram**

Proces algoritmu a jeho jednotlivé kroky zobrazuje diagramn na obr. 4.4



Obrázek 4.4: Schéma algoritmu víceobjektového sledování

- **Další funkcionality algoritmu**

Dosud byla nastíněna základní funkce algoritmu, která pokrývá značnou část vlastností potřebných ke splnění zadaného úkolu. Jevy, se kterými je algoritmus schopný počítat:

- Algoritmus se vypořádá s nejistotou asociace dat
- Umí se náležitě vypořádat s měřeními přicházejícími v nepravidelných časových intervalech
- Umí sledovat více objektů a přiřazovat objektům filtry dle jejich tříd
- Je schopný provádět sledování i pro objekty, které jsou například okludovány a nevyžaduje tedy měření každého objektu v každém kroku

- Umí se vypořádat s více měřeními téhož objektu během jednoho kroku

Jsou tu však i jevy, se kterými algoritmus pracovat neumí nebo pouze do omezené míry:

- Algoritmus nezahrnuje specifickou situaci, kdy objekt opustí scénu a může se na ni opět vrátit
- Problematickou částí algoritmu je inicializace filtrů — Jakmile je filtr jednou inicializován (je možné, že mylně), těžko se na základě nových měření upraví jeho odhad do správné pozice. Dva různé filtry sledující tutéž třídu mohou například splynout a sledovat zároveň jeden objekt. Nicméně je i možné, že se nakonec vlivem stochasticity částic rozdělí a každý začne sledovat jiný objekt.

Filtr je tedy třeba rozšířit o metody, které dané problémy řeší.

Opuštění scény objektem

V reálné situaci snímá kamera pouze určitý prostor a objekt se může během sledování dostat mimo něj. Žádoucí je tuto informaci do sledování zahrnout a držet informaci o tom, kolik objektů dané třídy se v daném okamžiku nachází na scéně a kolik mimo ni.

Toto lze zařídit tak, že pokud již inicializovaný objekt opustí scénu (nachází se poblíž okraje scény a poté od něj nedostáváme žádné měření), tuto informaci si zapamatujeme a budeme vyčkávat, dokud se opět měření této třídy neobjeví na okraji scény, kdy filtr můžeme k tomuto měření opět přiřadit.

Určení, zda objekt opustil scénu, můžeme provést následovně:

- Využijeme přítomnosti *nulové asociační hypotézy* u částic — pokud například predikujeme pozici objektu již mimo scénu a nedostaneme poblíž žádné měření dané třídy, převládne u asociace dat nulová hypotéza (měření ostatních objektů se budou nacházet ve větší vzdálenosti) a odhad stavu se tak již vyskytne mimo scénu.
- Pokud se odhad stavu objektu vyskytne mimo scénu, tuto informaci si zapamatujeme a dále filtr můžeme deinicializovat (stačí ponechat jeho částice na místě a neprovádět s ním žádné výpočetní operace)

Pokud máme informaci o tom, že se objekt některé třídy nachází mimo scénu, znovu inicializujeme filtr až poté, co se objeví nové měření dané třídy v blízkosti okraje scény. Tuto inicializaci můžeme opět provést tak, že si zkontrolujeme, zda dané měření na kraji scény se nenachází ve vzdálenosti $d < d_{max}$ vůči odhadu jiného objektu stejné třídy.

Problém inicializace

Problém inicializace spočívá zejména v tom, že celá povaha algoritmu je velmi založená na kontinuální pravděpodobnosti. Během asociace dat se nikde neprovádí diskrétní rozlišení, zda jde o tu či onu hypotézu. Každá hypotéza se projeví pouze v *míře* jejího vlivu na celkový stav filtru a předpokládá se, že nakonec se filtr i za cenu poněkud nižší přesnosti bude držet hypotézy správné.

Naproti tomu způsob, jak jsou filtry inicializovány, je prováděn právě skrz diskrétní rozlišení, a to dokonce v jednom kroku. Pokud tedy proběhne inicializace špatně, těžko se poté nastaví na správnou hypotézu.

Sledování orientace

Dosud jsme se v práci věnovali pouze sledování pozice, nikoliv orientace. Zde nastává otázka, jakým způsobem asociovat data orientace.

Je zřejmé, že asociace dat u orientace je závislá na asociaci dat u pozice. Pokud bychom explicitně přiřazovali jednotlivá měření pozic k jednotlivým filtrům, stejnou asociaci dat bychom mohli jednoduše provést i pro orientaci.

V našem algoritmu však není asociace dat explicitně počítána. Využít informace o asociaci dat z pozice lze však *korespondenčním* přístupem, který je v této práci navržen. Tímto přístupem může být zároveň zlepšena i samotná asociace dat pozičních filtrů.

Korespondenční asociace

Základní myšlenka korespondenční asociace spočívá ve využití informaci o naměřené hodnotě pozice k asociaci dat orientace a naopak.

Pokud dostaneme měření $\mathbf{z}_{k,j}$ společně s $\mathbf{q}_{k,j}$, můžeme spočítat, kterému filtru toto měření nejpravděpodobněji přísluší (porovnáních sum vah $w^{(l)}_{k,i,j}$ přičtených částicím jednotlivých filtrů).

Pokud přiřazení měření pozičnímu i orientačnímu filtru navzájem *koresponduje*, zvyšuje se tak pravděpodobnost, že dané měření opravdu přísluší filtru sledujícímu daným objekt. Tuto skutečnost můžeme například uplatnit tak, že přidané váhy ještě vynásobíme koeficientem korespondence $c_{cor} > 1$.

Pokud přiřazení spolu navzájem nekorespondují, pravděpodobnost asociace jak pozičního, tak orientačního měření naopak klesá. Tuto informaci můžeme uplatnit opět tak, že je vynásobíme koeficientem korespondence, ten však bude mít hodnotu $c_{cor} < 1$.

Toto je pouze jeden ze způsobů, jakým informace o korespondenci využít. Bylo by možné ještě zkoumat míru korespondence, či pravděpodobnost přiřazení měření jiným pozičním nebo orientačním filtrům a tuto informaci do výpočtu zahrnout.

Pro naše účely však stačí popsaný výpočet, neboť tak plní zadaný úkol za cenu nižší výpočtové náročnosti.

4.5 Evaluace víceobjektového sledování

Evaluace víceobjektového sledování probíhá stejným způsobem jako evaluace samotné filtrace. Je nutné:

- Vygenerovat vhodnou sadu dat — princip generování dat je popsán v kap. 4.2.1 a podrobněji způsob generování jednotlivých trajektorií ve scénáři je popsán v 3.9.1.
- Uložit data do správného formátu — při použití třídy *RandomizedSearchCV* je třeba využít 2D pole. Pokud chceme využít principu korespondenční asociace popsaného v kapitole kap. 4.4.5, musíme složit pole $\mathbf{X}$ a X_{rot}, stejně tak $\mathbf{Y}$ a $\mathbf{Y_{rot}}$ dohromady.
- Určit přibližný rozsah parametrů filtrů i samotného algoritmu sledování (například určit váhu nulové asociace $w_{k,i,0}$, průměrný počet měření dané třídy na krok k či konstanty korespondence c_{cor}.
- K určení těchto parametrů a konstant je potřeba algoritmus nejdříve manuálně testovat a ladit ve vizuálním prostředí. Jeho spolehlivá funkce je tedy prioritní úkol při snaze vyhodnotit fungování algoritmu.

Bohužel se ukázalo, že rozsah práce převyšoval očekávání. Samotná část filtrace a dále podpůrné funkcionality umožňující analýzu činnosti filtrů, generování trajektorií a správné formátování dat zabrala příliš času na to, aby mohl být vypracován algoritmus víceobjektového sledování s vlastnostmi popsanými v kap. 4.1.

Scénář byl tedy zjednodušen:

- Měření pocházela z pravidelných intervalů po $0.2s$. Bylo tak možné navázat na strukturu implementace u filtrace.
- Scénář obsahoval měření všech objektů v každém kroku.
- Objekty se vždy vyskytují na scéně

Dále se při implementaci algoritmu JPDA vyskytly komplikace, jejichž řešení by opět zabralo hodně času. Algoritmus byl sice otestován, ale jeho výsledky nebyly dobré (viz 5.2.

Rychle tak byl implementován jednoduchý asociační algoritmus GNN: v každém kroku se nové měření asociovalo tomu filtru, jehož poslední odhad stavu byl nejblíže tomuto měření.

5 Experimenty

5.1 Filtrace

Testovací data

Vstupní sada testovacích dat byla složena z celkem 24 trajektorií (s pozičními i orientačními daty). Jednalo se o 6 trajektorií každého ze 4 typů (viz parametry zadání kap. 3.1.1).

Hodnoty měření prostorové trajektorie byly v každé souřadnici zatíženy Gaussovským šumem se standardní odchylkou o velikosti *1,75cm*, *2cm* a *2,25cm*. Každá z těchto tří hodnot směrodatných odchylek byla vždy použita pro 2 trajektorie od každého typu.

Měření orientační trajektorie byla pro každý koeficient kvaternionu (a, b, c, d) zašuměna Gaussovským šumem o velikosti $\sigma_q = 8°$.

Ostatní hodnoty parametrů trajektorií lze vidět v tabulce tab. 5.1.

Název proměnné	Značení	Hodnota
Směrodatná odchylka šumu pozice	σ_z	$1,75cm$;$2cm$; $2,25cm$
Směrodatná odchylka šumu orientace	σ_q	$8°$
Maximální úhlová rychlost	ω_m	$25° \cdot s^{-1}$
Hranice pracovního prostoru	$\mathbf{B_w}$	$x\epsilon < -2; 2 > m, y\epsilon < -2; 2 > m, z\epsilon < 0; 4 > m$
Časový interval mezi měřeními	dt	$0,2s$
Průměrná rychlost objektu	v	$0,2m \cdot s^{-1}$
Střední hodnota klíčových bodů	λ_p	3 (Úsečky), 4 (Spline)
Limit vzdálenosti bodů	d_{min}	$0,15m$ (Úsečky), $0,35m$ (Spline)
Minimální poloměr	r_{min}	$0,2m$
Maximální poloměr	r_{max}	$2m$

Tabulka 5.1: Hodnoty parametrů pro generátory trajektorie

Způsob testování

Testování proběhlo pomocí třídy *RandomizedSearchCV()*, která otestovala výkon každého filtru pro náhodně vybrané parametry z rovnoměrných rozdělení v zadaných intervalech.

Kombinace parametrů pro každý filtr byly zprvu určeny na základě ručního upravování hodnot a zkoumání vlivu těchto úprav na výkon filtru. K tomu byla využita implementovaná vizualizace, která sloužila k získání hrubého odhadu vhodných a nevhodných hodnot parametrů.

Dále byl pro přesnější určení vhodných hodnot proveden testovací *RandomizedSearchCV* o menším počtu iterací (konkrétně 200). Tento test vybíral hodnoty parametrů z širších intervalů. Díky tomu bylo možné z výsledků testu zúžit rozpětí intervalů, vyloučit tak nevhodné hodnoty (dle špatných výsledků) a naopak zdůraznit vhodné hodnoty (dle dobrých výsledků).

Ukázalo se například, že některé parametry ovlivňují výsledek velmi silně (například σ_p u PFGH, některé naopak mají spíše zanedbatelný vliv (například $\sigma_\mathbf{Q}$ u KPF). Pro finální

testování byl u parametrů, které měly na výsledek větší vliv, zúžen interval náhodného výběru kolem hodnot, které měly dobré výsledky. Naopak u parametrů, které na výsledek příliš velký vliv neměly, byl ponechán poměrně široký interval náhodného výběru, aby byl její výběr rozmanitější a tím se mohla vybrat hodnota, která se dobře doplňuje s ostatními parametry.

Intervaly parametrů jsou vypsány v kap. 5.1.3 v tabulkách 5.2 až 5.9.

- **Testovací parametry filtrů**

Kalmanův filtr

Název parametru	Značení	Hodnoty
SO procesního šumu	σ_p	$[0,1; 0,45]\ (m)$
SO šumu měření	σ_z	$[0,008; 0,025]\ (m)$
Maximální hodnota ε	ε_{max}	$[0,2; 2,85]$
Škálovací faktor pro **Q**	Q_{scale}	$[10; 2000]$

Tabulka 5.2: Hodnoty parametrů Kalmanova filtru

Částicový filtr g-h

Název parametru	Značení	Hodnoty
Počet částic	N	$[600, 1000, 1500, 2500]$
SO procesního šumu	σ_p	$[0,01; 0,035]\ (m)$
SO šumu měření	σ_z	$[0,01; 0,035]\ (m)$
Koeficient g	g	$[0,3; 1]$
Koeficient h	h	$[0; 1]$

Tabulka 5.3: Hodnoty parametrů částicového filtru g-h

Částicový filtr s rychlostí

Název parametru	Značení	Hodnoty
Počet částic	N	$[600, 1000, 1500, 2500]$
SO procesního šumu	σ_p	$[0,01; 0,5](m)$
SO šumu měření	σ_z	$[0,0075; 0,025](m)$
SO procesního šumu rychlosti	σ_v	$[0,01; 0,375](m \cdot s^{-1})$
Konstanta pro reziduál	g_r	$[0; 1]$

Tabulka 5.4: Hodnoty parametrů částicového filtru s rychlostí

KPF

Název parametru	Značení	Hodnoty
Počet částic	N	$[600, 1000, 1500, 2500]$
SO procesního šumu	σ_p	$[0,004; 0,025](m)$
SO šumu měření	σ_z	$[0,0175; 0,04](m)$
SO procesního šumu Kalmanova filtru	σ_Q	$[0,02; 1,5](m \cdot s^{-1})$
SO šumu měření Kalmanova filtru	σ_R	$[0,025; 0,1](m \cdot s^{-1})$
Konstanta vlivu zrychlení na procesní šum	c_a	$[0,04; 0,85]$

Tabulka 5.5: Hodnoty parametrů KPF

Kalmanův filtr (orientace)

Název parametru	Značení	Hodnoty
SO procesního šumu	σ_p	$[0,3; 0,7](m)$
SO šumu měření	σ_z	$[0,004; 0,06](m)$
Maximální hodnota ε	ε_{max}	$[0,05; 1,2]$
Škálovací faktor pro $\mathbf{Q}$	Q_{scale}	$[10; 3500]$

Tabulka 5.6: Hodnoty parametrů Kalmanova filtru (orientace)

Částicový filtr g-h (orientace)

Název parametru	Značení	Hodnoty
Počet částic	N	$[600, 1000, 1500, 2500]$
SO procesního šumu	σ_p	$[0,2; 0,6]\ (m)$
SO šumu měření	σ_z	$[0,14; 0,325]\ (m)$
Koeficient g	g	$[0,4; 1]$
Koeficient h	h	$[0; 1]$

Tabulka 5.7: Hodnoty parametrů částicového filtru g-h (orientace)

Částicový filtr s rychlostí (orientace)

Název parametru	Značení	Hodnoty
Počet částic	N	$[600, 1000, 1500, 2500]$
SO procesního šumu	σ_p	$[0,2; 0,46](m)$
SO šumu měření	σ_z	$[0,275; 0,6](m)$
SO procesního šumu rychlosti	σ_v	$[0,15; 0,7](m \cdot s^{-1})$
Konstanta pro reziduál	g_r	$[0; 1]$

Tabulka 5.8: Hodnoty parametrů částicového filtru s rychlostí (orientace)

KPF (orientace)

Název parametru	Značení	Hodnoty
Počet částic	N	$[600, 1000, 1500, 2500]$
SO procesního šumu	σ_p	$[0,1; 0,375](m)$
SO šumu měření	σ_z	$[0,26; 0,5](m)$
SO procesního šumu Kalmanova filtru	σ_Q	$[0,25; 3](m \cdot s^{-1})$
SO šumu měření Kalmanova filtru	σ_R	$[0,25; 0,3](m \cdot s^{-1})$
Konstanta vlivu zrychlení na procesní šum	c_a	$[0,7; 2,35]$

Tabulka 5.9: Hodnoty parametrů KPF (orientace)

- **Výsledky filtrace**

Sada dat byla rozdělena na dvě poloviny:

- Lineární trajektorie — posloupnosti úseček s konstantní rychlostí a posloupnosti úseček s proměnlivou rychlostí
- Nelineární trajektorie — kružnice a spline

Pro každý filtr (celkem 8) a sadu trajektorií bylo provedeno 2880 iterací *RandomizedSearchCV()*, celkem tedy byla provedena celá filtrace sady trajektorií $2 \cdot 8 \cdot 2880 = 46080$-krát.

Nejlepší výsledky

V tabulkách 5.10, 5.11, 5.12 a 5.13 jsou uvedeny jednotlivé filtry s jejich nejlepšími výsledky.

Filtr	MSE	Průměrný čas filtrace
KPF	$0,19376mm^2$	$0,9772s$
PFGH	$0,1984mm^2$	$0,985s$
Kalmanův filtr	$0,20436mm^2$	$0,0389s$
PF s rychlostí	$0,20456mm^2$	$0,7642s$

Tabulka 5.10: Výsledky pozičních filtrů pro lineární trajektorie

Filtr	MSE	Průměrný čas filtrace
KPF	$0,15275mm^2$	$1,06813s$
PF s rychlostí	$0,1712mm^2$	$1,6617s$
Kalmanův filtr	$0,17522mm^2$	$0,05599s$
PFGH	$0,17900mm^2$	$1,0730s$

Tabulka 5.11: Výsledky pozičních filtrů pro nelineární trajektorie

Filtr	MSE	Průměrný čas filtrace
KPF	$0,0012725\|\|\mathbf{q}\|\|^2$	$0,73s$
PFGH	$0,001725\|\|\mathbf{q}\|\|^2$	$0,75s$
PF s rychlostí	$0,0018508\|\|\mathbf{q}\|\|^2$	$1,5351s$
Kalmanův filtr	$0,003715\|\|\mathbf{q}\|\|^2$	$0,04618s$

Tabulka 5.12: Výsledky orientačních filtrů pro lineární trajektorie

Filtr	MSE	Průměrný čas filtrace
KPF	$0,0019388\|\|\mathbf{q}\|\|^2$	$1,3981s$
PF s rychlostí	$0,002445\|\|\mathbf{q}\|\|^2$	$2,1528s$
PFGH	$0,002608\|\|\mathbf{q}\|\|^2$	$1,47s$
Kalmanův filtr	$0,003989\|\|\mathbf{q}\|\|^2$	$0,06643s$

Tabulka 5.13: Výsledky orientačních filtrů pro nelineární trajektorie

Odchylka orientace byla pro každý krok filtru počítána jakožto $d\mathbf{q} = ||\hat{\mathbf{q}}_k - \mathbf{q}_k||^2$, proto je její použitá jednotka rovna $||\mathbf{q}||^\mathbf{2}$.

Z výsledků můžeme rovnou porovnat jednotlivé filtry. Vítěz s ohledem na přesnost je jednoznačně KPF a vítěz s ohledem na rychlost jednoznačně Kalmanův filtr. Pokud by předmětem práce bylo sledovat pouze jeden objekt, bylo by použití Kalmanova filtru s dalšími případnými vylepšeními tou nejlepší volbou. Rozdíl přesnosti není (zejména pro měření pozice) tak velký, zatímco doba výpočtu je nesrovnatelně nižší. Částicový filtr však má svůj význam při sledování více objektů.

Další výsledek, který není překvapující, je poměr výkonu PFGH a částicového filtru s rychlostí. Pro lineární trajektorie se ukazuje jako lepší filtr PFGH, zatímco u nelineárních trajektorií vykazuje lepší výkony částicový filtr s rychlostí. To je nejspíše dáno tím, že vlastnost částic držet v sobě i údaj o rychlosti u PF s rychlostí umožňuje větší adaptabilitu na nelineární trajektorie.

Na druhou stranu je však PF s rychlostí náročnější na výpočet, což je pro naše účely velká překážka.

Nicméně nejlepšího výsledku z hlediska přesnosti dosahuje pro všechny typy trajektorií KPF, a tak bude použit pro účely sledování více objektů. Dobrý výkon filtru v obou typech trajektorií můžeme připsat jeho vlastnosti adaptivního filtrování.

Nejlepší výsledky pro menší počet částic

Jelikož náš filtr má prioritizovat i rychlost výpočtu, je dobré zhodnotit nejlepší výsledky filtrů pro menší počet částic, u kterých je kratší výpočetní čas - konkrétně 600.

Filtr	MSE	Průměrný čas filtrace
KPF	$0,19842mm^2$	$0,6611s$
PFGH	$0,20345mm^2$	$0,51s$
PF s rychlostí	$0,22705mm^2$	$0,6161s$

Tabulka 5.14: Výsledky pozičních filtrů pro lineární trajektorie — 600 částic

Filtr	MSE	Průměrný čas filtrace
KPF	$0,15908mm^2$	$0,9260s$
PF s rychlostí	$0,18064mm^2$	$0,8679s$
PFGH	$0,18322mm^2$	$0,7280s$

Tabulka 5.15: Výsledky pozičních filtrů pro nelineární trajektorie — 600 částic

Filtr	MSE	Průměrný čas filtrace
KPF	$0,001773\|\|\mathbf{q}\|\|^2$	$0,6632s$
PFGH	$0,001982\|\|\mathbf{q}\|\|^2$	$0,5735s$
PF s rychlostí	$0,002327\|\|\mathbf{q}\|\|^2$	$0,6554s$

Tabulka 5.16: Výsledky orientačních filtrů pro lineární trajektorie — 600 částic

Filtr	MSE	Průměrný čas filtrace
KPF	$0,0019388\|\|\mathbf{q}\|\|^2$	$1,3981s$
PFGH	$0,003020\|\|\mathbf{q}\|\|^2$	$0,8870s$
PF s rychlostí	$0,003196\|\|\mathbf{q}\|\|^2$	$0,9428s$

Tabulka 5.17: Výsledky orientačních filtrů pro nelineární trajektorie — 600 částic

Porovnáním tabulek 5.10 až 5.13 s tabulkami 5.14 až 5.17 si můžeme všimnout, že lze poměrně dost snížit dobu trvání filtrace, aniž by byl výkon filtr příliš ovlivněn. Například u PFGH se při snížení času téměř na polovinu zhoršila přesnost jen o $2,54\%$.

Dále se ukazuje, že částicový filtr s rychlostí funguje podstatně lépe při vyšším počtu částic. To je vidět v tab. 5.17, kdy se při použití menšího počtu částic výsledek zhoršil tak, že je již horší než u filtru PFGH.

- **Výsledné parametry filtrů**

Zbývá zhodnotit, pro které kombinace parametrů fungují jednotlivé filtry nejlépe. 10 nejlepších kombinací parametrů pro každý filtr na každém typu trajektorií je zobrazeno v tabulkách A.2 a A.3

Některé poznatky, které můžeme z tabulek pro **poziční** filtry vyčíst, jsou:

- Pro Kalmanův filtr má překvapivě procesního šum **Q** hodnotu vyšší u lineárních trajektorií než u nelineárních. Na druhou stranu je však u nelineárních trajektorií výrazně vyšší hodnota Q_{scale}, což značí větší efekt adaptivního filtrování.
- Podobný jev lze pozorovat u PFGH, kde je pro lineární trajektorie přítomen vyšší procesní šum. To však může být vykompenzováno nižší hodnotou h, což způsobuje rychlejší reakci na změnu zrychlení objektu.
- U částicového filtru s rychlostí vidíme u nelineárních trajektorií vyšší procesní i rychlostní šum, zároveň také menší hodnotu g_r. Všechny tyto hodnoty znamenají větší reaktivitu vůči změnám trajektorie, což odpovídá lepším výsledkům u trajektorií nelineárních. Dále také můžeme vidět poměrně vysokou hodnotu σ_z, vzhledem k tomu, že skutečný šum měření u trajektorie byl roven $0,02m$.

- U KPF vidíme u nelineární trajektorie opět menší procesní šum, což se zdá nečekané. Nejspíš je tento jev vykompenzován podstatně vyšší hodnotou c_a, která dle naměřeného zrychlení škáluje právě procesní šum částic. Filtr je tak u nelineárních trajektorií více adaptabilní.

Poznatky, které můžeme vyčíst z tabulek pro **orientační** filtry, jsou:

- Nelineární trajektorie byly pravděpodobně nedopatřením vytvořeny s vyšší celkovou hodnotou šumu měření. Odpovídá tomu obecně vyšší optimální hodnota parametru σ_z u orientačních filtrů (s výjimkou Kalmanova filtru), včetně vyšších hodnot procesního šumu. Dále je také známkou této skutečnosti fakt, že nejlepší výsledky orientačních filtrů jsou u nelineárních trajektorií podstatně horší, ačkoliv jsou v podstatě stejné (nelinearita trajektorií se projevuje pouze v pozici)
- U filtru PFGH můžeme pozorovat velmi vysoké hodnoty g, což značí silnou konzistenci v odhadu (úhlové) rychlosti. Jelikož je trajektorie lineární extrapolací orientací mezi dvěma kvaterniony, výsledek tomu odpovídá.

5.2 Víceobjektové sledování

Vzhledem k rozsahu přípravné práce pro ohodnocení víceobjektového sledování a zaměření na základní část sledování (filtraci), nezbyl v této práci prostor pro systmatické otestování výkonu algoritmu víceobjektového sledování. Byl vytvořen skript pro vizualizaci, v němž je možné průběh sledování zobrazit a vyhodnotit předběžné výsledky.

Výsledky s asociací na bázi JPDA

Algoritmus více-objektového sledování na bázi asociace JPDA byl testován na vygenerovaném scénáři dvou trajektorií zobrazeném v příloze na obr. A.4.

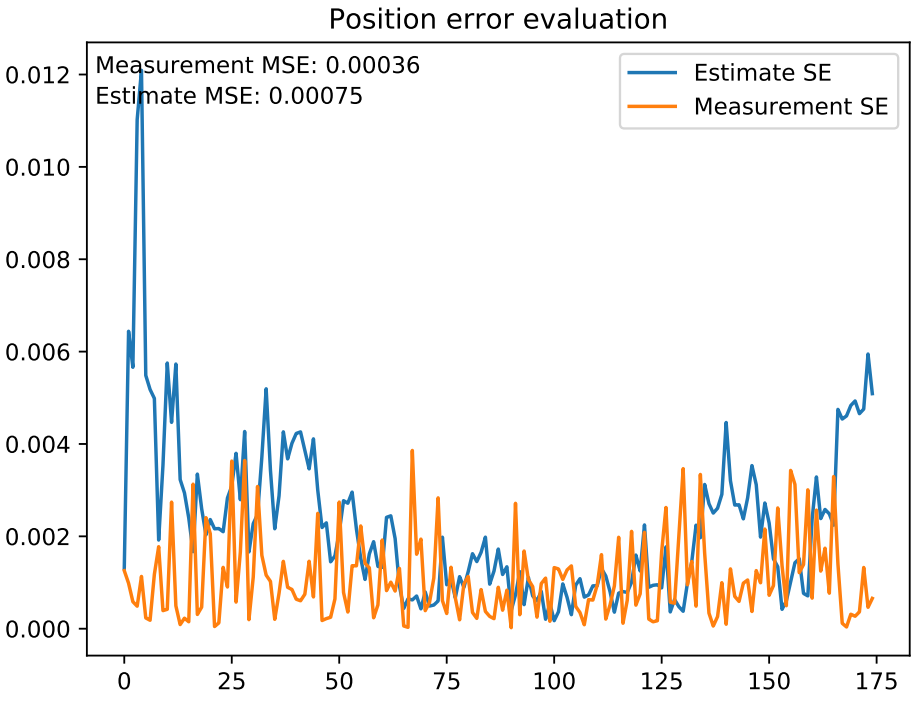
Algoritmus byl testován pro hodnotu matice tříd

$$\mathbf{C} = \begin{bmatrix} 1 & 0,1 & 0,1 & 0,1 \\ 0,1 & 1 & 0,1 & 0,1 \\ 0,1 & 0,1 & 1 & 0,1 \\ 0,1 & 0,1 & 0,1 & 1 \end{bmatrix} \tag{5.1}$$

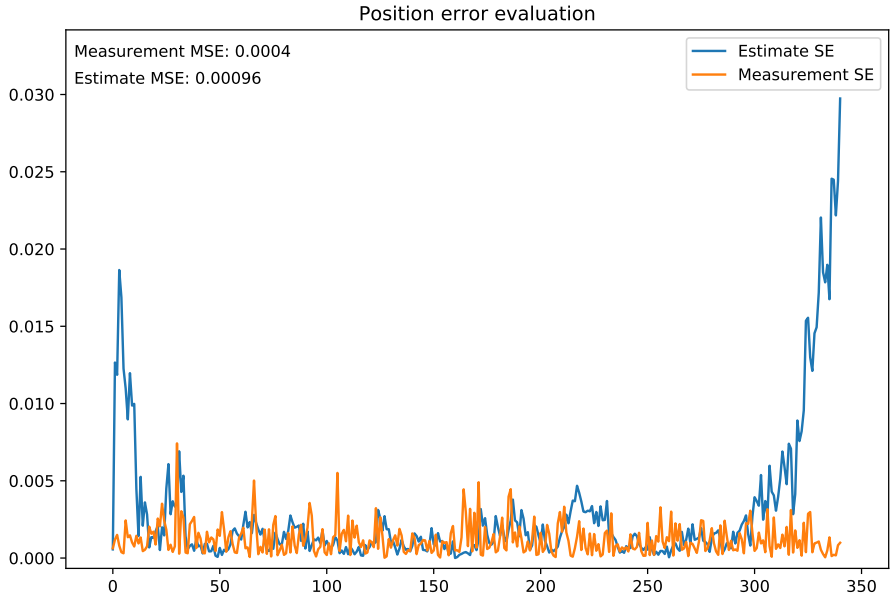
Hodnoty MSE a zároveň grafy odchylek měření a odhadu z filtru pro jednotlivé trajektorie jsou zobrazeny na obrázcích 5.1 a 5.2

Výsledky s asociací GNN

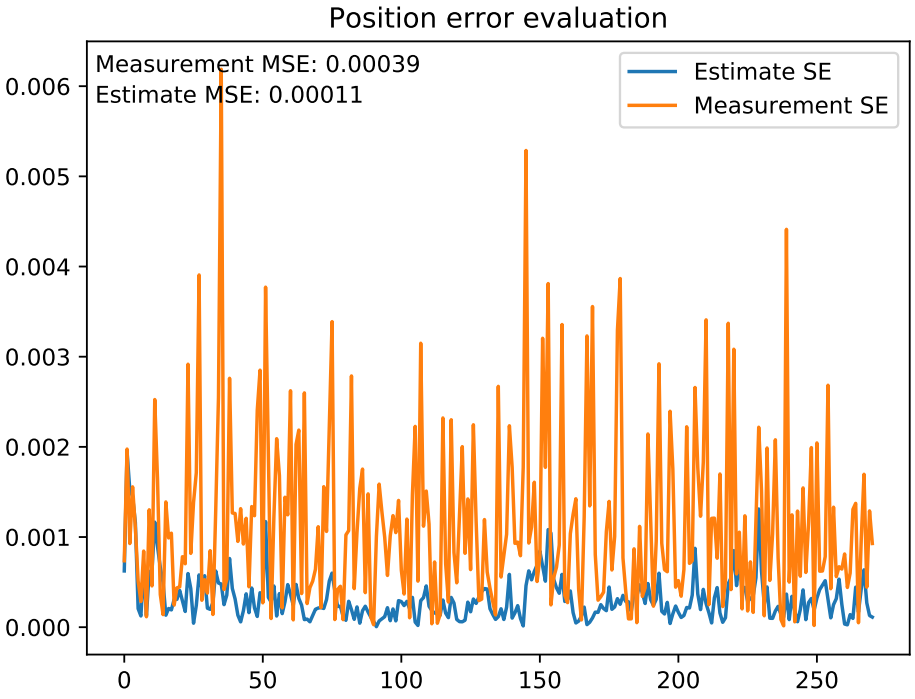
Výsledky s jednodušším systémem asociace dat GNN byly demonstrovány na vygenerovaném scénáři se čtyřmi trajektoriemi zobrazenými v příloze na A.5. Hodnoty MSE a zároveň grafy odchylek měření a odhadu z filtru pro jednotlivé trajektorie jsou zobrazeny na obrázcích 5.3 až 5.6



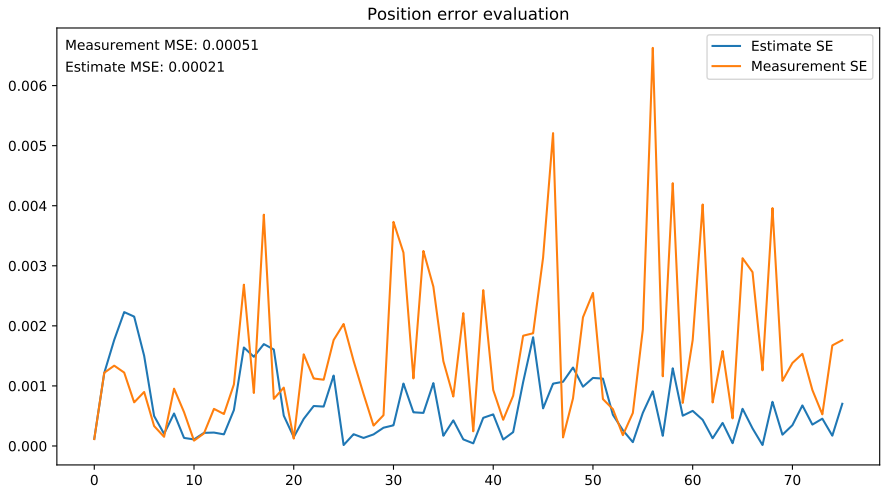
Obrázek 5.1: Výsledky MOT pro algoritmus JPDA trajektorie 1



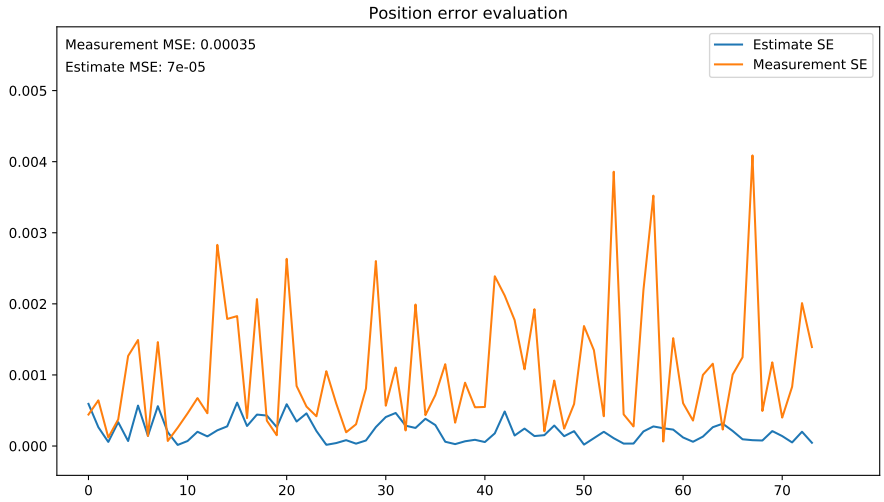
Obrázek 5.2: Výsledky MOT pro algoritmus JPDA trajektorie 2



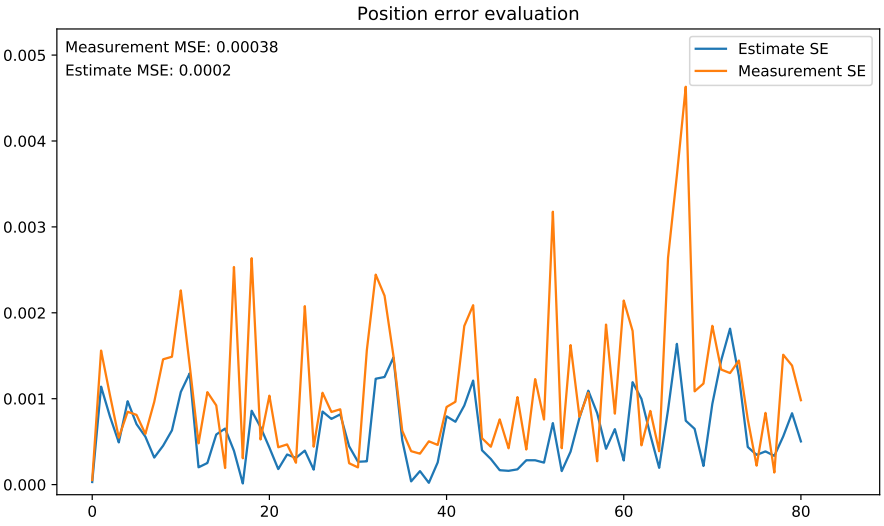
Obrázek 5.3: Výsledky víceobjektového sledování trajektorie 1



Obrázek 5.4: Výsledky víceobjektového sledování trajektorie 2



Obrázek 5.5: Výsledky víceobjektového sledování trajektorie 2



Obrázek 5.6: Výsledky víceobjektového sledování trajektorie 2

6 Diskuze

6.1 Více-objektové sledování

Jak již bylo zmíněno v předchozích kapitolách, nepodařilo se plně implementovat navržený a popsaný algoritmus MOT na bázi JPDA. Jeho výsledky jsou horší než šum měření a bylo by potřeba implementaci přepracovat. Je vysoce pravděpodobné, že nefunkčnost algoritmu (respektive jeho špatná výkonnost) je způsobena chybou v implementovaném programu. Bohužel nezbyl v práci čas na to, aby tato chyba byla nalezena a opravena.

Při asociaci dat ve stylu GNN se podařilo v testovaném scénáři dosáhnout kvalitních výsledků srovnatelných s výsledky získanými při řešení samotné filtrace pózy jednoho objektu. Je však třeba zdůraznit, že pokud by byl scénář složitější (měření v nepravidelných intervalech, trajektorie blíže sebe, simulace okluze objektů), GNN by provedl několik špatných asociacích a jeho chyba by se výrazně zvětšila.

I přes nedokončenou finální implementaci algoritmu byl vytvořen funkční vizualizační skript, díky kterému lze na práci při vývoji sledovacích algoritmů navázat.

7 Závěr

V této práci byla teoreticky prozkoumána problematika filtrace 6D pózy objektu spolu s více-objektovým sledováním. Hlavním cílem práce byla tvorba sledovacího algoritmu založeném na použití částicového filtru.

Algoritmus měl fungovat nezávisle na použitém detektoru, aby mohl být univerzálněji využit. Sledován měl být známý počet objektů určitých tříd na scéně.

Jako součást práce byly vytvořeny dva hlavní skripty umožňujících testování filtrů, generování trajektorií, vizualizaci procesu filtrace a systematickou evaluaci filtrovacích algoritmů. V dalších souborech byly naprogramovány modifikovatelné třídy celkem tří částicových filtrů a dále rozšiřující třída již hotové implementace Kalmanova filtru, která sloužila ke srovnání výkonu částicových filtrů s obyčejným Kalmanovým filtrem.

Dále byly vytvořeny třídy pro generátory různých typů trajektorií, které bylo možné zobrazit ve vizuálním prostředí a uložit je pro další použití. Tímto způsobem vzniklo rozhraní, na které je možné navázat implementací dalších filtrů či sledovacích algoritmů, generováním jiných typů trajektorií či laděním filtrů stávajích.

Jednotlivé filtry byly systematicky otestovány na datech simulující reálnou situaci, kdy dostáváme zašuměná měření. Hodnocen byl čas trvání filtrace a přesnost ve smyslu průměrné čtvercové chyby oproti vygenerovaným ground truth datům. Skrze testování byly identifikovány a vypsány vhodné parametry pro jednotlivé filtry.

Nejlepších výsledků dosáhl částicový filtr s Kalmanovým filtrem pro určení rychlosti (KPF), který byl následně použit pro více-objektové sledování. Minimální průměrná čtvercová chyba dosáhla hodnoty $0,000152m^2$ v porovnání s čtvercovou chybou šumu měření, která dosahovala $0,00053m^2$.

Byly prozkoumány možnosti návrhu algoritmů pro více-objektové sledování a pro tuto práci byl vybrán algoritmus na bázi JPDA, jehož vlastnosti se doplňují s vlastnostmi částicového filtru. Algoritmus byl implementován a částečně otestován ve vizuálním prostředí, které muselo být spolu s formátem generovaných a ukládaných dat o trajektoriích pro více-objektové sledování rozšířeno.

Zásadní část práce představovala implementace a testování samotné filtrace. Dále pak implementace rozhraní, jehož funkcemi jsou:

- Generování trajektorií, včetně jejich vizualizace a systému ukládání a nahrávání
- Vizualizace procesu filtrace včetně částic
- Okamžité vyhodnocení výsledků po skončení filtrace

Ukázalo se, že rozsah této části zásadně převýšil očekávání a bohužel nebyla dokončena implementace algoritmu víceobjektového sledování na bázi algoritmu JPDA.

Algoritmus byl implementován, nicméně pravděpodobně vlivem implementační chyby nevykazoval dobré výsledky V závěru práce byl proto implementován jednodušší algoritmus GNN, jehož výsledky na testovacím scénáři vykazovaly chybu měření srovnatelnou s chybou při samotné filtraci, konkrétně mezi $7 \cdot 10^{-g}m^2$ a $0,00021m^2$.

8 Zdroje

[1] Y. Zhang, T. Wang a X. Zhang, „MOTRv2: Bootstrapping End-to-End Multi-Object Tracking by Pretrained Object Detectors,“ in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, s. 22 056–22 065.

[2] T.-T. Le, T.-S. Le, Y.-R. Chen, J. Vidal a C.-Y. Lin, „6D pose estimation with combined deep learning and 3D vision techniques for a fast and accurate object grasping,“ *Robotics and Autonomous Systems*, roč. 141, s. 103 775, 2021, ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2021.103775. URL: https://www.sciencedirect.com/science/article/pii/S0921889021000609.

[3] M. Vavrecka, N. Sokovnin, M. Mejdrechova a G. Sejnova, „MyGym: Modular Toolkit for Visuomotor Robotic Tasks,“ in *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, 2021, s. 279–283. DOI: 10.1109/ICTAI52525.2021.00046.

[4] Y. Bukschat a M. Vetter, *EfficientPose: An efficient, accurate and scalable end-to-end 6D multi object pose estimation approach*, 2020. arXiv: 2011.04307 [cs.CV].

[5] K. Granström, M. Fatemi a L. Svensson, „Poisson Multi-Bernoulli Mixture Conjugate Prior for Multiple Extended Target Filtering,“ *IEEE Transactions on Aerospace and Electronic Systems*, roč. 56, č. 1, s. 208–225, 2020. DOI: 10.1109/TAES.2019.2920220.

[6] L. Svensson a K. Granström, „ChalmersX: Multi-Object Tracking for Automotive Systems,“ *edX, Videos*, 2019. URL: https://www.youtube.com/@multipleobjecttracking1226.

[7] M. O. Tracking, *GNN - Basic Idea, Prediction and Update*, 2019. URL: https://www.youtube.com/watch?v=MDMNsQJl6-Q&list=PLadnyz93xCLiCBQq1105j5Jeqi1Q6wjoJ&index=20&ab_channel=MultipleObjectTracking.

[8] M. O. Tracking, *JPDA - Examples*, 2019. URL: https://www.youtube.com/watch?v=v2P2VjUPnl0&list=PLadnyz93xCLiCBQq1105j5Jeqi1Q6wjoJ&index=24&ab_channel=MultipleObjectTracking.

[9] M. O. Tracking, *Track Oriented-MHT - Prediction and Update*, 2019. URL: https://www.youtube.com/watch?v=HafGfdr0qbo&list=PLadnyz93xCLiCBQq1105j5Jeqi1Q6wjoJ&index=29&ab_channel=MultipleObjectTracking.

[10] F. García-Fernández, J. L. Williams, K. Granström a L. Svensson, „Poisson Multi-Bernoulli Mixture Filter: Direct Derivation and Implementation,“ *IEEE Transactions on Aerospace and Electronic Systems*, roč. 54, č. 4, s. 1883–1901, 2018. DOI: 10.1109/TAES.2018.2805153.

[11] R. Labbe, https://github.com/rlabbe/Kalmanand-Bayesian-Filters-in-Python, 2015.

[12] W. Luo, X. Zhao a T. Kim, „Multiple Object Tracking: A Review,“ *CoRR*, roč. abs/1409.7618, 2014. arXiv: 1409.7618. URL: http://arxiv.org/abs/1409.7618.

[13] L. Zhang a L. van der Maaten, „Preserving Structure in Model-Free Tracking,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, roč. 36, č. 4, s. 756–769, 2014. DOI: 10.1109/TPAMI.2013.221.

[14] L. Buitinck, G. Louppe, M. Blondel et al., „API design for machine learning software: experiences from the scikit-learn project,“ in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, s. 108–122.

[15] L. Svensson, D. Svensson, M. Guerriero a P. Willett, „Set JPDA Filter for Multitarget Tracking,“ *IEEE Transactions on Signal Processing*, roč. 59, č. 10, s. 4677–4691, 2011. DOI: 10.1109/TSP.2011.2161294.

[16] B. Song, T.-Y. Jeng, E. Staudt a A. K. Roy-Chowdhury, „A Stochastic Graph Evolution Framework for Robust Multi-target Tracking,“ in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos a N. Paragios, ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, s. 605–619, ISBN: 978-3-642-15549-9.

[17] S. Oh, S. Russell a S. Sastry, „Markov Chain Monte Carlo Data Association for Multi-Target Tracking,“ *IEEE Transactions on Automatic Control*, roč. 54, č. 3, s. 481–497, 2009. DOI: 10.1109/TAC.2009.2012975.

[18] B. Bose, X. Wang a E. Grimson, „Multi-class object tracking algorithm that handles fragmentation and grouping,“ in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007, s. 1–8. DOI: 10.1109/CVPR.2007.383175.

[19] M. Yang, T. Yu a Y. Wu, „Game-Theoretic Multiple Target Tracking,“ in *2007 IEEE 11th International Conference on Computer Vision*, 2007, s. 1–8. DOI: 10.1109/ICCV.2007.4408942.

[20] M. Jaward, L. Mihaylova, N. Canagarajah a D. Bull, „Multiple object tracking using particle filters,“ in *2006 IEEE Aerospace Conference*, 2006, 8 pp.–. DOI: 10.1109/AERO.2006.1655926.

[21] C. Chang, R. Ansari a A. Khokhar, „Multiple object tracking with kernel particle filter,“ in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, sv. 1, 2005, 566–573 vol. 1. DOI: 10.1109/CVPR.2005.243.

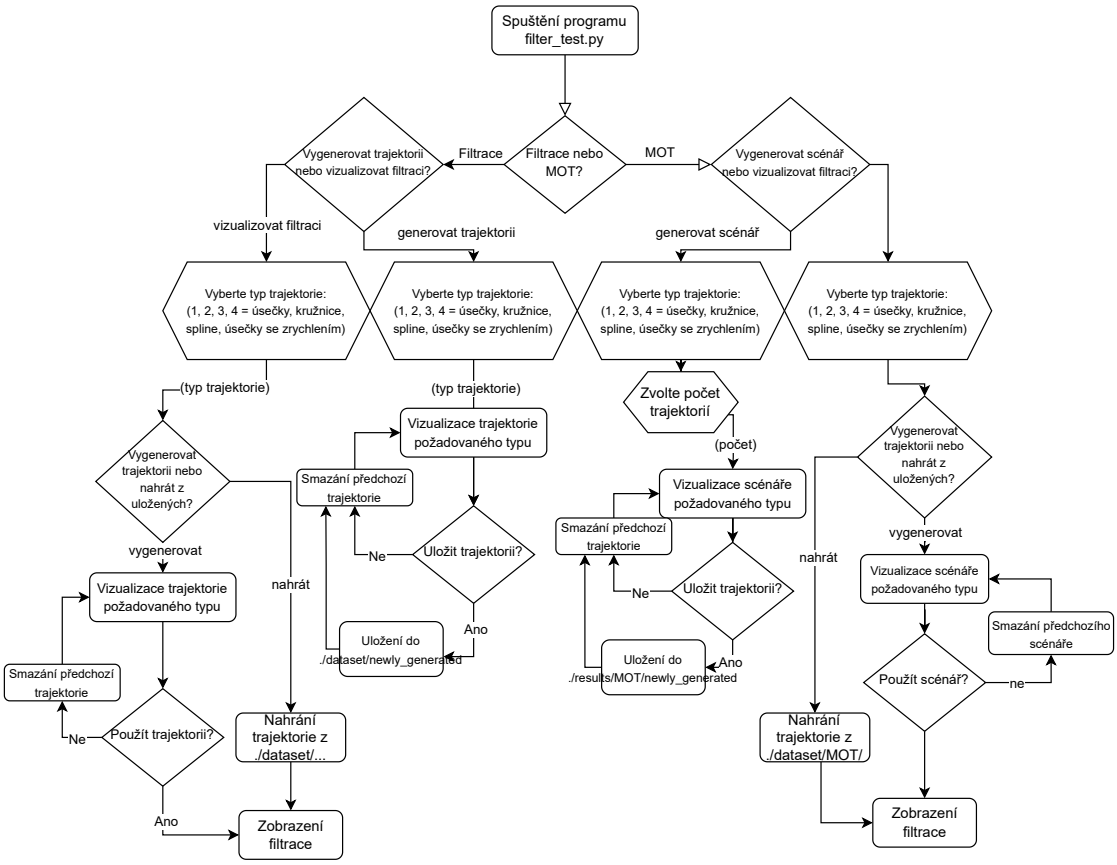
[22] Z. Khan, T. Balch a F. Dellaert, „An MCMC-Based Particle Filter for Tracking Multiple Interacting Targets,“ in *Computer Vision - ECCV 2004*, T. Pajdla a J. Matas, ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, s. 279–290, ISBN: 978-3-540-24673-2.

[23] T.-J. Cham a J. Rehg, „A multiple hypothesis approach to figure tracking,“ in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, sv. 2, 1999, 239–244 Vol. 2. DOI: 10.1109/CVPR.1999.784636.

[24] G. Welch, G. Bishop et al., „An introduction to the Kalman filter,“ 1995.

[25] D. Reid, „An algorithm for tracking multiple targets,“ *IEEE Transactions on Automatic Control*, roč. 24, č. 6, s. 843–854, 1979. DOI: 10.1109/TAC.1979.1102177.

A Appendix A



Obrázek A.1: Návodový diagram pro spouštění scriptu filter_test.py

Kalmanův filtr pozice, lineární trajektorie				
	Q	R	Maximální epsilon	Škálovač Q
	0,31301	0,01542	0,68288	68,78443
	0,41333	0,01433	0,57494	90,17414
	0,30246	0,01258	0,39904	30,60206
	0,36407	0,01455	0,48519	1545,44460
	0,41458	0,01988	0,33802	719,22944
	0,35836	0,01357	0,50167	1573,99353
	0,39740	0,01904	0,36174	633,01832
	0,38204	0,01557	0,42916	1808,50033
	0,31352	0,01467	0,46335	484,26781
	0,37583	0,01835	0,38897	752,37025
Průměr	0,36346	0,01580	0,46250	770,63849

PF s rychlostí pozice, lineární trajektorie				
Počet částic	Procesní šum	Šum rychlosti	Šum měření	G reziduálu
1000	0,01919	0,01679	0,03003	0,65024
1000	0,02905	0,03581	0,04919	0,65668
2500	0,10984	0,04892	0,04898	0,55134
2500	0,12115	0,05154	0,07011	0,60485
2500	0,11768	0,04304	0,04941	0,44461
2500	0,02618	0,03062	0,04739	0,76544
2500	0,07132	0,01348	0,04193	0,94048
2500	0,09831	0,03702	0,03692	0,27826
2500	0,01049	0,09055	0,06876	0,45976
2500	0,25017	0,04389	0,06124	0,48130
Průměr	0,08534	0,04117	0,05040	0,58330

Kalmanův filtr pozice, nelineární trajektorie				
	Q	R	Maximální epsilon	Škálovač Q
	0,15535	0,02056	0,31691	1727,18280
	0,14356	0,01738	0,37920	776,13816
	0,10584	0,00877	0,96149	1786,69074
	0,11074	0,01016	0,83000	1881,22281
	0,11796	0,01257	0,41674	617,25515
	0,14482	0,02437	0,26907	1278,80714
	0,14437	0,01593	0,38329	1865,93302
	0,11014	0,00805	1,06842	1227,25562
	0,14897	0,01503	0,38109	1113,89184
	0,16831	0,01745	0,35496	1333,26524
Průměr	0,13501	0,01503	0,53612	1360,76425

PF s rychlostí pozice, nelineární trajektorie				
Počet částic	Procesní šum	Šum rychlosti	Šum měření	G reziduálu
2500	0,04290	0,07989	0,03854	0,13543
2500	0,06747	0,03722	0,03279	0,19002
2500	0,04732	0,03632	0,02939	0,29737
2500	0,05602	0,08778	0,06642	0,29778
2500	0,05450	0,02229	0,04849	0,69407
2500	0,14552	0,09521	0,03573	0,15000
2500	0,27181	0,06645	0,07033	0,25267
2500	0,04758	0,25163	0,07227	0,08418
2500	0,29961	0,06703	0,06236	0,18623
1500	0,20591	0,06311	0,03302	0,11289
Průměr	0,11475	0,08265	0,05070	0,25419

PFGH pozice, lineární trajektorie				
Počet částic	Procesní šum	Šum měření	g	h
2500	0,01854	0,03547	0,76956	0,80480
2500	0,01967	0,02788	0,76335	0,85063
1000	0,02220	0,03531	0,73250	0,88763
2500	0,01804	0,02881	0,80569	0,81353
1500	0,02026	0,03252	0,73354	0,85176
1000	0,01969	0,03208	0,72930	0,88987
1000	0,01874	0,03313	0,82143	0,49935
1000	0,01490	0,02500	0,74675	0,73836
600	0,01893	0,03501	0,62275	0,97467
1500	0,02218	0,03364	0,84191	0,71887
Průměr	0,01932	0,03189	0,75668	0,80295

KPF pozice, lineární trajektorie					
Počet částic	Procesní šum	Šum měření	Q	R	Konstanta a
1500	0,01293	0,02867	0,27599	0,08935	0,10511
2500	0,01495	0,03660	0,22632	0,09294	0,23005
2500	0,01303	0,02858	0,21173	0,08102	0,21732
2500	0,01709	0,02954	0,12032	0,07809	0,06103
2500	0,01229	0,03249	0,16028	0,07372	0,26034
2500	0,01040	0,02666	0,31522	0,06977	0,17356
2500	0,01851	0,03810	0,14703	0,06124	0,23235
600	0,01644	0,02924	0,17913	0,09667	0,09627
2500	0,01297	0,02894	0,26871	0,08676	0,17960
1000	0,01312	0,02836	0,20946	0,07614	0,22741
Průměr	0,01417	0,03072	0,21142	0,08057	0,17830

PFGH pozice, nelineární trajektorie				
Počet částic	Procesní šum	Šum měření	g	h
1500	0,01966	0,03569	0,83506	0,73476
1500	0,01664	0,03693	0,72970	0,97112
2500	0,01228	0,02802	0,81086	0,74852
1000	0,01654	0,03480	0,85253	0,53884
600	0,01914	0,03628	0,83922	0,79125
1500	0,01637	0,03710	0,85580	0,46799
2500	0,01575	0,03357	0,60531	0,97847
1500	0,01883	0,03370	0,68603	0,93944
1500	0,01161	0,03331	0,80841	0,11278
1500	0,01343	0,03371	0,78975	0,13778
Průměr	0,01602	0,03431	0,78127	0,64209

KPF pozice, nelineární trajektorie					
Počet částic	Procesní šum	Šum měření	Q	R	Konstanta a
1000	0,00906	0,03413	0,02065	0,08052	0,34430
1000	0,00865	0,03073	0,03844	0,08406	0,25556
2500	0,01057	0,03118	0,02691	0,03399	0,24163
1000	0,00842	0,03106	0,05960	0,04318	0,23034
1500	0,01080	0,03899	0,03239	0,09959	0,37544
2500	0,00535	0,03377	0,02627	0,08264	0,40911
2500	0,00716	0,03809	0,08496	0,05228	0,31179
1500	0,01304	0,03612	0,05041	0,08655	0,28371
1500	0,00421	0,02977	0,07966	0,09997	0,35629
2500	0,00873	0,03346	0,08835	0,08994	0,36638
Průměr	0,00860	0,03373	0,05076	0,07527	0,31746

Obrázek A.2: Nejlepší kombinace parametrů filtrů pozice

Kalmanův filtr orientace, lineární trajektorie			
Q	R	Maximální epsilon	Škálovač Q
0,64890	0,00402	0,79792	2221,98353
0,66218	0,00410	0,41876	3357,38341
0,63957	0,00414	0,65741	681,98804
0,62199	0,00405	0,16139	209,15076
0,66649	0,00445	0,80127	545,71660
0,64434	0,00449	1,01336	844,21207
0,56728	0,00409	0,81311	889,97246
0,58183	0,00420	1,00315	1404,38304
0,58131	0,00421	0,53394	188,06765
0,69088	0,00497	0,54796	2603,61797
0,63048	0,00427	0,67483	1294,64755

KPF orientace, lineární trajektorie					
Počet částic	Procesní šum	Šum měření	Q	R	Konstanta a
1000	0,15827	0,49876	0,75958	2,62421	0,86890
1500	0,12494	0,29864	0,80650	2,46365	1,33259
2500	0,15174	0,33292	0,33778	2,39569	1,29661
2500	0,14051	0,37768	1,07764	1,82248	1,03377
2500	0,29771	0,45654	0,85129	1,49512	1,57369
2500	0,12555	0,34929	0,27090	2,73887	2,11040
2500	0,13869	0,33756	0,52069	2,38329	1,33175
2500	0,14471	0,33381	0,77360	2,84238	2,06325
2500	0,10109	0,35065	0,50915	2,15297	1,82231
2500	0,17696	0,42328	0,63358	2,49918	2,27857
Průměr	0,15602	0,37591	0,65407	2,34178	1,57118

Kalmanův filtr orientace, nelineární trajektorie			
Q	R	Maximální epsilon	Škálovač Q
0,64848	0,00409	0,61806	508,61442
0,65177	0,00422	0,61873	2299,43938
0,60854	0,00401	0,16871	614,90691
0,60601	0,00424	0,06843	3243,74612
0,54050	0,00402	0,74339	1141,34121
0,64782	0,00474	0,39915	1456,83583
0,57948	0,00432	0,92168	2429,63734
0,63668	0,00475	0,74119	2316,39510
0,54981	0,00419	0,91479	684,25458
0,57175	0,00435	0,43843	1971,18120
0,60408	0,00429	0,56326	1666,63521

KPF orientace, nelineární trajektorie					
Počet částic	Procesní šum	Šum měření	Q	R	Konstanta a
1500	0,17900	0,40198	0,28306	2,92552	2,24679
1500	0,16888	0,39744	0,90794	2,60361	1,90023
2500	0,13118	0,45286	1,10358	0,89928	2,05638
1000	0,12115	0,35484	0,36733	0,89108	1,73699
2500	0,17404	0,45741	1,54068	2,20629	2,18877
2500	0,18293	0,36504	0,83419	0,46896	0,83319
2500	0,17926	0,35182	0,26092	2,08618	1,27524
1500	0,12350	0,39511	0,94703	0,87164	2,06163
2500	0,17606	0,45722	0,44649	0,88555	0,91070
2500	0,21856	0,47635	1,80801	1,42027	2,07240
Průměr	0,16546	0,41101	0,84992	1,52584	1,72823

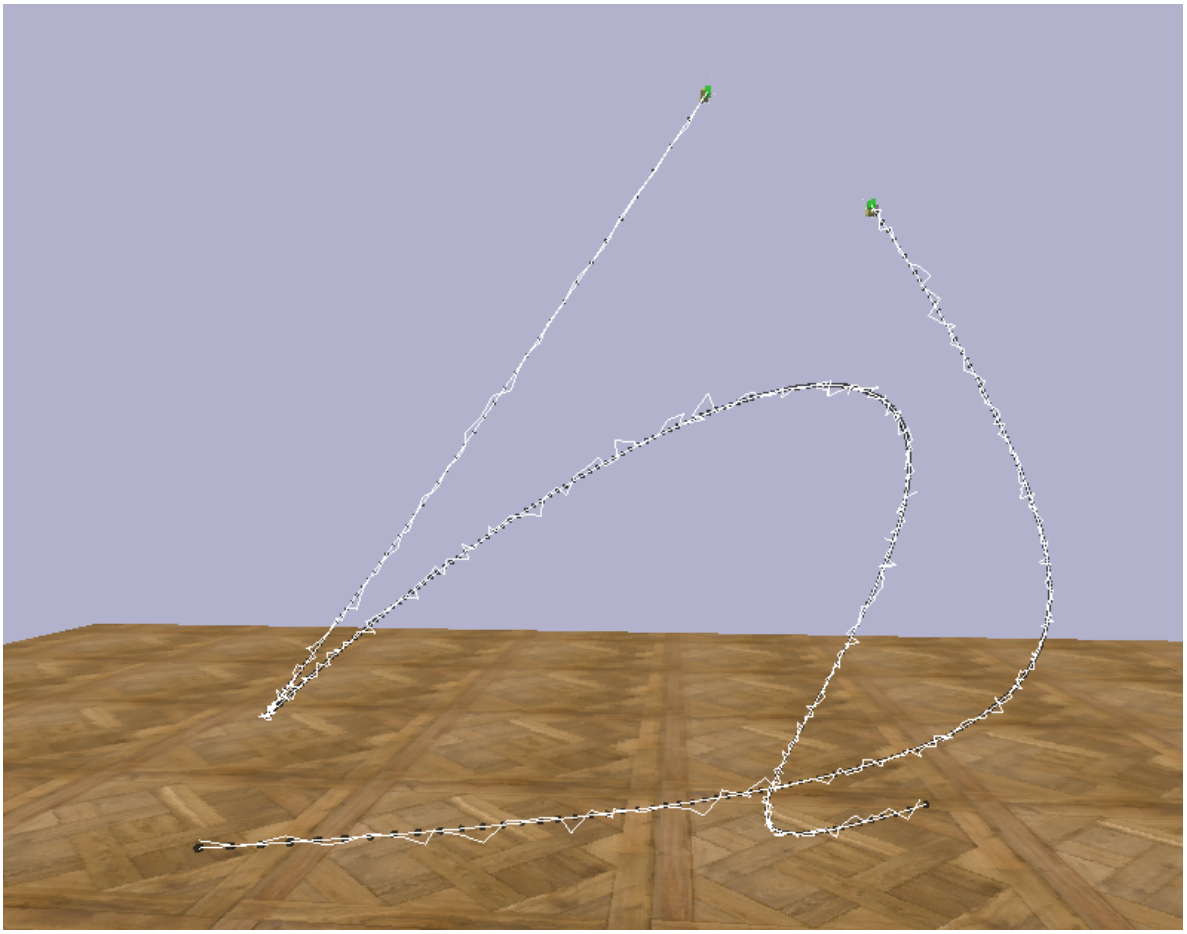
PF s rychlostí orientace, lineární trajektorie				
Počet části	Procesní š	Šum rychlosti	Šum měření	G reziduálu
2500	0,27103	0,27669	0,42210	0,41320
2500	0,24559	0,15950	0,30084	0,29292
2500	0,24779	0,16076	0,34145	0,66603
2500	0,30878	0,15810	0,40606	0,38040
2500	0,30938	0,21276	0,37194	0,45004
2500	0,27190	0,19114	0,28855	0,19152
2500	0,25733	0,17779	0,28126	0,29878
1500	0,33529	0,24348	0,45578	0,56277
2500	0,32632	0,20866	0,46611	0,97449
2500	0,32031	0,20023	0,42529	0,45752
Průměr	0,28937	0,19891	0,37594	0,46877

PFGH orientace, lineární trajektorie				
Počet částic	Procesní šum	Šum měření	g	h
1500	0,22440	0,31533	0,99205	0,95552
2500	0,23872	0,28409	0,96991	0,91244
1500	0,20029	0,25813	0,95904	0,56748
2500	0,24639	0,30409	0,98129	0,83554
2500	0,27085	0,29028	0,95970	0,50236
1500	0,23350	0,31145	0,92523	0,07358
2500	0,20558	0,26396	0,89896	0,28834
2500	0,26061	0,28874	0,99125	0,39049
1500	0,21627	0,26147	0,97652	0,27503
1000	0,20259	0,32144	0,81086	0,86601
Průměr	0,22992	0,28990	0,94648	0,56668

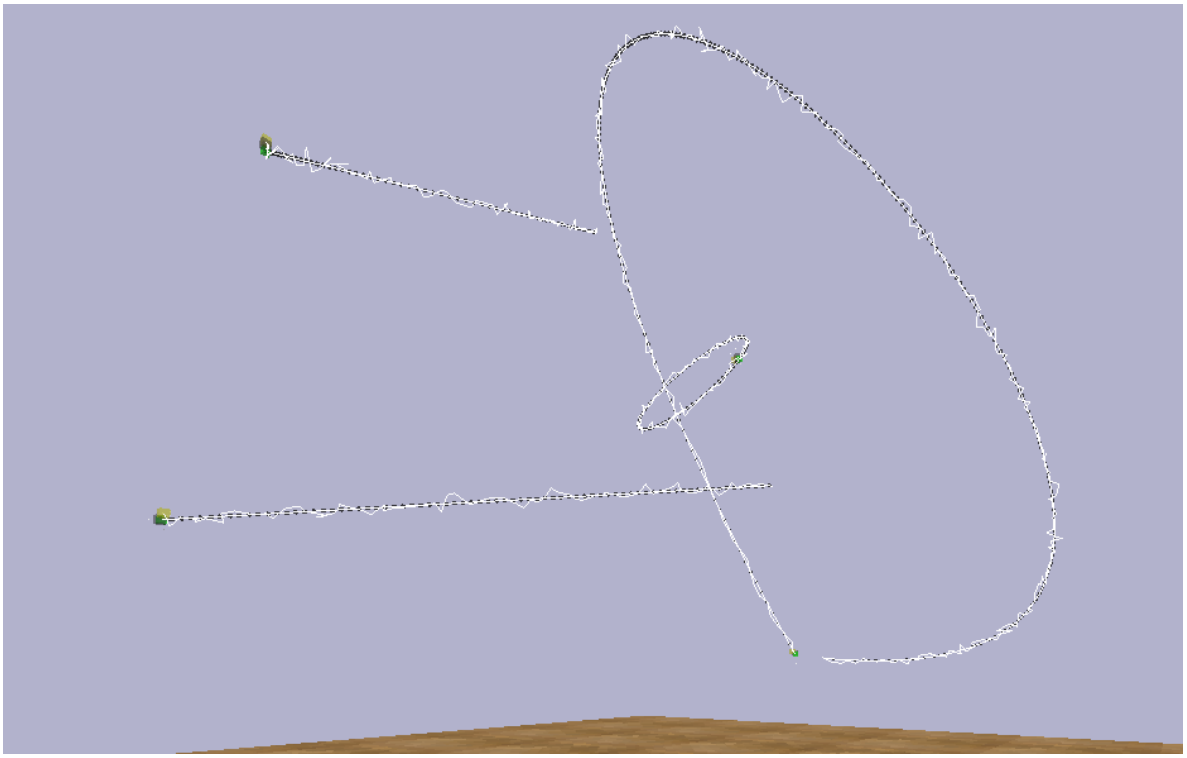
PF s rychlostí orientace, nelineární trajektorie				
Počet části	Procesní š	Šum rychlosti	Šum měření	G reziduálu
2500	0,24128	0,25289	0,40249	0,92884
2500	0,32847	0,16596	0,37632	0,57071
2500	0,30105	0,27699	0,42585	0,81424
2500	0,35264	0,24406	0,45961	0,99019
2500	0,37294	0,19612	0,40098	0,71444
2500	0,21569	0,58270	0,49671	0,65723
2500	0,25087	0,41553	0,49158	0,82379
2500	0,33483	0,22571	0,42795	0,81484
2500	0,37187	0,27790	0,31087	0,84087
2500	0,38003	0,31882	0,38668	0,67503
Průměr	0,31497	0,29567	0,41790	0,78302

PFGH orientace, nelineární trajektorie				
Počet částic	Procesní šum	Šum měření	g	h
2500	0,21185	0,29900	0,88643	0,04120
1500	0,29664	0,29864	0,95147	0,81183
2500	0,31581	0,31979	0,92433	0,73178
1500	0,39193	0,31452	0,94091	0,52239
2500	0,46089	0,32024	0,97007	0,05229
2500	0,35203	0,32098	0,99624	0,12057
2500	0,37083	0,31844	0,98158	0,28919
2500	0,41813	0,30548	0,86051	0,90258
2500	0,21180	0,30500	0,96256	0,29913
2500	0,20565	0,31880	0,92019	0,40325
Průměr	0,32356	0,31209	0,93943	0,41742

Obrázek A.3: Nejlepší kombinace filtrů orientace



Obrázek A.4: Testovací trajektorie pro více-objektové sledování JPDA



Obrázek A.5: Testovací trajektorie pro více-objektové sledování GNN

Odkaz na github repozitář: https://github.com/incognite-lab/myGym/tree/particle_filter