

Sveučilište u Zagrebu

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Vjekoslav Diklić

CLASSIFICATION OF VEHICLE TRAJECTORY LOGICAL SEGMENTS FOR SCENARIO-BASED SIMULATION TESTING

DOCTORAL THESIS

Zagreb, 2026.



University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Vjekoslav Diklić

CLASSIFICATION OF VEHICLE TRAJECTORY LOGICAL SEGMENTS FOR SCENARIO-BASED SIMULATION TESTING

DOCTORAL THESIS

Supervisor: Professor Ivan Marković, PhD

Zagreb, 2026.



Sveučilište u Zagrebu

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Vjekoslav Diklić

KLASFIKACIJA LOGIČKIH SEGMENATA TRAJEKTORIJE VOZILA ZA SIMULACIJSKA TESTIRANJA ZASNOVANA NA SCENARIJIMA

DOKTORSKI RAD

Mentor: prof. dr. sc. Ivan Marković

Zagreb, 2026.

Doctoral thesis was written at the University of Zagreb, Faculty of Electrical Engineering and Computing, Departement of Control and Computer Engineering.

Supervisor: Professor Ivan Marković, PhD

Thesis contains 116 pages

ABOUT THE SUPERVISOR

Ivan Marković was born in Osijek, Croatia in 1985. He finished a math-oriented high school in Osijek in 2003 and received the master and PhD degrees in Electrical Engineering under the mentorship of Prof. Ivan Petrović from the University of Zagreb Faculty of Electrical Engineering and Computing (FER), Zagreb, Croatia in 2008 and 2014, respectively. In 2008 he was employed by S.C.A.N. Ltd. where he worked on developing control algorithms and supervision interfaces for terminal automation systems. In 2009. he joined FER and the Department of Control and Computer Engineering as a research assistant where he enrolled the PhD studies in the same year, and since 2023 works as a Full Professor. In 2009 and 2012 he was a member of the National Organizing Committee and in 2022 and 2023 the Programme Committee Chair of international scientific conferences. In 2018. he became the deputy Editor-In-Chief of the journal "Automatika – Journal for Control, Measurement, Electronics, Computing and Communications" and since 2025 he serves as the Editor-In-Chief. In collaboration with Siemens Croatia Inc.. he has been a course trainer for programmes developed for industry participants and in 2018 he received the recognition for his contribution in advancing education for SIMATIC automation systems. He actively participated as a researcher or work package leader in 11 national and 9 international research projects. During 2013 and 2014 he was a visiting researcher in the laboratory led by Prof. Francois Chaumette at INRIA Rennes-Bretagne Atlantique u Rennesu, France. He published 31 journal papers and more than 70 conference papers in the fields of estimation theory, sensor fusion, and robot vision with applications to autonomous mobile robot and vehicle control. He is a member of the Institute of Electrical and Electronics Engineers (IEEE), Technical Committee for Robotics of the International Federation of Automatic Control (IFAC), and the Croatian Society for Communication, Computing, Electronics, Measurement and Control (KoREMA). In 2023. he received the National Science Award by the Republic of Croatia, in 2018 the young scientist award "Vera Johanides" of the Croatian Academy of Engineering, and in 2014 silver plaque "Josip Lončar" of FER for an outstanding doctoral dissertation.

O MENTORU

Ivan Marković je rođen 1985. godine u Osijeku. Treću gimnaziju u Osijeku završio je 2003. godine te je diplomirao i doktorirao u polju elektrotehnike pod mentorstvom prof. dr. sc. Ivana Petrovića na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva (FER), 2008. odnosno 2014. godine. Po završetku studija zapošljava se u tvrtki S.C.A.N. d.o.o. gdje radi na razvijanju algoritama upravljanja i nadzora u sustavima automatizacije naftnih terminala. U 2009. godini zapošljava se u Zavodu za automatiku i računalno inženjerstvo FER-a u zvanju znanstvenog novaka u suradničkom zvanju asistenta te iste godine upisuje i doktorski studi, a od 2023. godine radi u zvanju redovitog profesora. U 2009. i 2012. godini bio je član organizacijskog odbora, a u 2022. i 2023. predsjednik programskog odbora međunarodnih znanstvenih konferencija. Od 2018. godine je zamjenik glavnog i odgovornog urednika časopisa "Automatika – časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije", a od 2025. preuzima tu dužnost. U suradnji s tvrtkom Siemens Hrvatska d.d. bio predavač je na tečajevima za polaznike iz industrije te je 2018. godine primio priznanje za poseban doprinos unaprijeđenu nastave i edukaciji korištenjem SIMATIC automatizacijskih sustava. Aktivno je sudjelovao kao suradnik ili voditelj radnog paketa na 11 nacionalnih i 9 međunarodnih znanstvenih projekata. Tijekom 2013. i 2014. godine bio je gostujući istraživač u grupi prof. dr. sc. Francoisa Chaumettea na ustanovi INRIA Rennes-Bretagne Atlantique u Rennesu, Francuska. Objavio je 31 znanstveni rad u časopisima i više od 70 znanstvenih radova u zbornicima skupova u području estimacije, fuzije osjetila i robotskog vida s primjenom u upravljanju autonomnim mobilnim robotima i vozilima. Član je društva Institute of Electrical and Electronics Engineers (IEEE), tehničkog odbora za robotiku društva International Federation of Automatic Control (IFAC) i Hrvatskog društva za komunikacije, računarstvo, elektroniku, mjerenja i automatiku (KoREMA). Godine 2023. primio je Državnu nagradu za znanost Republike Hrvatske, 2018. nagradu mladom znanstveniku "Vera Johanides" Akademije tehničkih znanosti Hrvatske te 2014. srebrnu plaketu "Josip Lončar" FER-a za posebno istaknutu doktorsku disertaciju.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor, Prof. Ivan Marković, for his guidance, patience, and invaluable support throughout the course of this work.

I am sincerely thankful to my colleagues and friends whose contributions, encouragement, and support greatly enriched this work.

Above all, I wish to thank my family and loved ones for their constant encouragement, understanding, and unconditional support, which gave me strength throughout this journey.

ABSTRACT

This thesis investigates methods and evaluation metrics for converting recorded vehicle trajectories into logical segments that represent vehicle actions, enabling automated reconstruction of driving scenarios for scenario-based simulation testing. The overall goal is to transform raw trajectories derived from real-world data into higher-level logic scenario representations that support parameter variation. Two methods for lateral vehicle action classification are addressed. The first method is based on polynomial modelling in a curvilinear coordinate system, where lateral motion is approximated using a third-degree polynomial. The second method applies a neural-network-based classifier, providing a data-driven approach to lateral vehicle action recognition. In addition to classification, the thesis introduces a simulation-based trajectory filtering and vehicle parameter estimation procedure based on extended moving horizon estimation and model predictive control principles. The introduced filtering method can be used as a preparation step for disturbed input trajectories, reducing noise and irregularities and improving physical plausibility before classification and scenario reconstruction. To evaluate classification results in a manner that is meaningful for logic scenario generation, this thesis proposes dedicated metrics for lateral action classification. These metrics quantify the accuracy of detected action start and end points, the spatial offset of detected maneuvers relative to reference centerlines, and the temporal continuity of classification within individual actions. Together, the proposed classification methods, preprocessing procedure, and evaluation metrics support reliable and scalable conversion of recorded driving data into logic scenarios, reducing manual effort and improving reproducibility in scenario-based simulation testing.

KEYWORDS: vehicle trajectory, logic segments, scenario-based testing, lane change classification, polynomial modelling, neural networks, moving horizon estimation, lane change, simulation segment, vehicle driving simulation, detection, scenario detection

KLASFIKACIJA LOGIČKIH SEGMENATA TRAJEKTORIJE VOZILA ZA SIMULACIJSKA TESTIRANJA ZASNOVANA NA SCENARIJIMA

Ovaj rad istražuje metode i metrike za pretvaranje snimljenih putanja vozila u logičke segmente koji opisuju radnje vozila, čime se omogućuje automatizirana rekonstrukcija scenarija vožnje za simulacijska testiranja zasnovana na scenarijima. Polazište rada je potreba za razvojem postupaka koji iz velikih skupova snimljenih podataka vožnje mogu izdvojiti informacije uporabive za izradu scenarija prikladnih za provjeru i validaciju suvremenih cestovnih vozila. Razvoj naprednih sustava pomoći vozaču i sustava automatizirane vožnje zahtijeva velik broj ponovljivih, raznolikih i dobro definiranih ispitnih slučajeva. Budući da je takve ispitne slučajeve teško, skupo i često nesigurno ostvariti isključivo stvarnim vožnjama, značajan se dio razvoja i provjere provodi u simulacijskom okruženju. Međutim, kvaliteta simulacijskog testiranja ovisi o kvaliteti scenarija koji se u simulaciji izvode. Scenariji moraju biti dovoljno reprezentativni da obuhvate stvarna prometna ponašanja, ali i dovoljno prilagodljivi da omogućuju promjenu parametara i sustavno proširivanje ispitnog prostora. Glavni cilj ovog doktorskog rada je pretvorba sirovih putanja dobivenih iz stvarnih podataka u logičke scenarije višeg stupnja apstrakcije koji omogućuju varijaciju parametara. Snimljene putanje vozila u izvornom obliku najčešće predstavljaju konkretne scenarije, odnosno točno određene vremenske zapise položaja vozila i drugih veličina koje opisuju gibanje. Takvi konkretni scenariji omogućuju ponavljanje zabilježene situacije, ali ne omogućuju jednostavno mijenjanje početnih uvjeta, trajanja radnji, brzina, udaljenosti između vozila, bočnih pomaka ili drugih parametara koji su važni za sustavno ispitivanje. Nasuprot tome, logički scenariji opisuju prometnu situaciju pomoću strukture i parametara koji se mogu mijenjati unutar zadanih raspona. Time se iz jedne snimljene putanje može dobiti opis koji nije vezan samo uz jedno ponavljanje, nego može služiti kao osnova za stvaranje većeg broja konkretnih scenarija. U tom se kontekstu putanja vozila ne promatra samo kao niz uzoraka položaja u vremenu, nego kao slijed radnji koje vozilo izvodi. Te radnje mogu biti bočne i uzdužne. Bočne radnje opisuju ponašanje vozila u odnosu na prometne trakove, primjerice zadržavanje prometne trake, promjenu prometne trake ulijevo i promjenu prometne trake udesno. Uzdužne radnje opisuju promjenu gibanja vozila u smjeru vožnje, primjerice ubrzavanje, usporavanje ili gibanje približno stalnom brzinom. Kombiniranjem bočnih i uzdužnih radnji moguće je snimljenu putanju podijeliti na logičke segmente. Svaki takav segment opisuje jedan dio gibanja vozila i može se koristiti kao element logičkog scenarija. Na mjestu na kojem dolazi do preklapanja bočnih i uzdužnih

radnji uvodi se novi segment. Na taj se način složena snimljena putanja pretvara u niz jednostavnijih, semantički određenih dijelova. Izvorni znanstveni doprinos rada očituje se u povezivanju više postupaka koji zajedno omogućuju automatiziranu pretvorbu putanja u scenarije. Ti postupci uključuju klasifikaciju bočnih radnji vozila, opis uzdužnog gibanja, formiranje logičkih segmenata, simulacijsko filtriranje putanja, estimaciju parametara vozila i razvoj metrika koje su prilagođene procjeni uporabljivosti rezultata za generiranje logičkih scenarija. Pritom se klasifikacija ne promatra samo kao problem dodjeljivanja oznaka pojedinačnim uzorcima, nego kao dio šireg postupka u kojem rezultat mora biti prikladan za izgradnju scenarija koji se može ponovno izvesti u simulaciji.

U radu se razmatra širi okvir simulacijskih testiranja zasnovanih na scenarijima. Takva testiranja predstavljaju jedan od ključnih pristupa za razvoj i validaciju funkcija povezanih s automatiziranom vožnjom jer omogućuju ponovljivu provjeru ponašanja vozila u unaprijed definiranim prometnim situacijama. Scenariji se mogu promatrati na više razina apstrakcije. Funkcijski scenariji opisuju prometnu situaciju općenitim jezikom i bez preciznih numeričkih vrijednosti. Logički scenariji definiraju strukturu situacije i raspone mogućih vrijednosti parametara. Konkretni scenariji sadrže točno određene vrijednosti svih parametara potrebnih za izvođenje simulacije. Snimljene putanje vozila prirodno odgovaraju konkretnim scenarijima jer sadrže točno zabilježene položaje i vremenske trenutke. Međutim, za sustavno simulacijsko testiranje korisnije je dobiti logički opis jer se iz njega mogu generirati različite varijacije istog osnovnog prometnog događaja. Ručna pretvorba snimljenih putanja u logičke scenarije moguća je, ali je vremenski zahtjevna, subjektivna i teško primjenjiva na velike skupove podataka. Velike baze putanja, prikupljene kamerama, senzorima ili simulacijama, mogu sadržavati tisuće vozila i velik broj manevara. Ako bi se svaki manevar morao ručno pregledati, označiti i pretvoriti u scenarij, takav bi postupak bio spor i nedovoljno skalabilan. Zbog toga je u ovom radu razvijen pristup koji nastoji automatizirati ključne korake pretvorbe. Ti koraci uključuju pripremu ulaznih putanja, klasifikaciju bočnih radnji, opis uzdužnog gibanja, formiranje logičkih segmenata, određivanje početnih uvjeta scenarija, filtriranje poremećenih putanja i vrednovanje rezultata pomoću metrika koje su prilagođene stvarnoj uporabi u generiranju logičkih scenarija. Teorijska osnova rada uključuje krivolinijski koordinatni sustav, polinomsko i splajnsko modeliranje, metode strojnog učenja, modeliranje vozila i upravljanje zasnovano na predikciji modela. Krivolinijski koordinatni sustav posebno je važan jer omogućuje opis položaja vozila u odnosu na cestu i prometne trakove. Umjesto promatranja putanje samo u globalnom Kartezijevu koordinatnom sustavu, položaj vozila može se opisati uzdužnom koordinatom duž ceste i bočnim odmakom od referentne linije. Takav opis prikladan je za prepoznavanje bočnih radnji, jer se promjena prometne trake očituje kao promjena bočnog položaja vozila u odnosu na prometne linije. Polinomsko modeliranje i modeliranje s pomoću splajnova primjenjuju se za opisivanje oblika putanje i smanjenje složenosti podataka. Snimljene putanje mogu sadržavati velik broj uzoraka, od kojih svaki opisuje položaj vozila u određenom vremenskom trenutku. Za potrebe logičkog scenarija nije nužno zadržati svaki uzorak, nego je potrebno izdvojiti jednostavniji opis koji zadržava bitna svojstva gibanja. Polinomski modeli omogućuju analitički opis pojedinih dijelova putanje, dok se postupci smanjenja broja čvorova mogu koristiti za prikaz uzdužnog gibanja pomoću manjeg broja segmenata. Time se omogućuje prijelaz iz podatkovno gustog zapisa prema kompaktnom opisu

prikladnom za simulaciju. U radu se također razmatraju metode strojnog učenja, osobito neuronske mreže, jer one omogućuju podatkovno vođen pristup prepoznavanju radnji vozila. Za razliku od metoda koje se oslanjaju na unaprijed definirana pravila, neuronske mreže uče obrasce iz označenih primjera. To je posebno korisno kada se oblik manevara razlikuje od idealiziranih pretpostavki ili kada su podaci narušeni šumom. Uz metode strojnog učenja, važnu ulogu ima i modeliranje vozila. Ako se snimljena putanja želi upotrijebiti za simulacijsko testiranje, potrebno je uzeti u obzir fizičku izvedivost gibanja. Zbog toga se u radu uvodi i upravljanje zasnovano na predikciji modela, koje se kasnije koristi za filtriranje putanja i estimaciju parametara vozila. U teorijskom se dijelu razmatraju i umjetno uvedeni poremećaji, šum i smetnje u podacima koji se koriste u eksperimentima. Snimljene putanje mogu biti narušene šumom, izdvojenim pogrešnim vrijednostima i nepravilnim uzorkovanjem. Takvi poremećaji mogu nastati zbog ograničenja senzora, pogrešaka detekcije, obrade videozapisa ili drugih mjernih nesigurnosti. Ako se takvi podaci izravno koriste za klasifikaciju i rekonstrukciju scenarija, pogreške se mogu prenijeti u konačni logički opis. Zbog toga je važno ispitati kako predložene metode rade pri različitim razinama kvalitete podataka i razviti postupke koji smanjuju utjecaj poremećaja.

Uz klasifikaciju, u radu je uveden simulacijski postupak filtriranja putanja i estimacije parametara vozila, zasnovan na proširenoj estimaciji pokretnog horizonta i principima upravljanja zasnovanog na predikciji modela. Svrha tog postupka je priprema ulaznih putanja prije klasifikacije i rekonstrukcije scenarija. Umjesto da se putanja samo matematički zagladi, predloženi pristup nastoji simulirati gibanje koje je u skladu s odabranim modelom vozila. To znači da filtrirana putanja ne smije biti samo glatka, nego mora biti fizički konzistentna. Takav pristup važan je jer simulacijski scenarij mora biti izvediv. Putanja koja sadrži nagle i fizički nemoguće promjene položaja, brzine ili ubrzanja može uzrokovati probleme pri ponovnom izvođenju u simulaciji. U postupku filtriranja ulazne putanje najprije se pripremaju za obradu. To uključuje usklađivanje vremenskih uzoraka, popunjavanje nedostajućih vrijednosti i izračun veličina koje nisu izravno dostupne, kao što su brzina, orijentacija i druge komponente stanja vozila. Nakon toga se koristi model vozila koji pokušava pratiti zadanu putanju uz zadana ograničenja. Proširena estimacija pokretnog horizonta omogućuje da se pri optimiranju uzme u obzir više uzastopnih vremenskih trenutaka, a ne samo trenutna mjerena točka. Time se postiže bolja vremenska dosljednost i smanjuje osjetljivost na lokalne pogreške u podacima. Upravljački ulazi i stanja vozila određuju se tako da rekonstruirano gibanje bude što bliže snimljenoj putanji, ali i u skladu s ograničenjima modela. Predložena metoda filtriranja može se koristiti kao pripremni korak za poremećene ulazne putanje, pri čemu se smanjuje utjecaj šuma i nepravilnosti te poboljšava fizička konzistentnost putanja prije klasifikacije i rekonstrukcije scenarija. Takvo filtriranje osobito je korisno kada se radi s putanjama dobivenima iz videozapisa, jer one mogu sadržavati male nepravilnosti koje nisu povezane sa stvarnim gibanjem vozila. Ako se takve nepravilnosti izravno koriste u klasifikaciji, mogu dovesti do niza pogrešnih kratkih segmenata, lažnih promjena radnji ili pomaka granica manevara. Filtriranjem se dobiva stabilniji ulaz za kasnije korake obrade. Budući da se obrada putanja za generiranje scenarija može provoditi izvan stvarnog vremena, veća računalna složenost ovakvog pristupa prihvatljiva je u zamjenu za bolju kvalitetu rezultata. Predloženi postupak uspoređuje se s klasičnim pristupima filtriranja i zaglađivanja, pri čemu je naglašena razlika između

matematičkog zaglađivanja signala i rekonstrukcije putanje koja mora biti izvediva za model vozila. Klasični filtri mogu smanjiti šum, ali ne moraju nužno osigurati da dobivena putanja odgovara fizički mogućem gibanju. U ovom radu putanja se rekonstruira tako da ju prati simulirani model vozila, čime se dobiva cjelovitiji zapis stanja i upravljačkih ulaza. Takav zapis može biti koristan ne samo za kasniju klasifikaciju, nego i za analizu ponašanja vozila te za pripremu scenarija koji se mogu izvesti u simulacijskom okruženju. Isti okvir omogućuje i estimaciju parametara vozila. U radu je kao primjer razmatrana estimacija mase vozila. Masa vozila obično nije dostupna u javnim skupovima putanja, ali može imati važan utjecaj na dinamičko ponašanje vozila. Procjena mase temelji se na ideji da će model s prikladnijom vrijednosti mase bolje rekonstruirati promatranu putanju od modela s neprikladnom vrijednosti. Usporedbom rezultata simulacije za različite pretpostavljene vrijednosti mase moguće je procijeniti koja vrijednost daje fizički najdosljednije slaganje s ulaznim podacima. Takav pristup pokazuje da se iz snimljenih putanja mogu izdvojiti i dodatne informacije o vozilu, a ne samo geometrijski opis gibanja. Procjena parametara vozila ipak ima svoja ograničenja. Da bi se neki parametar mogao pouzdano procijeniti, promatrana putanja mora sadržavati dovoljno informativnih dijelova gibanja. Ako vozilo dugo vozi približno stalnom brzinom i bez izraženih manevara, utjecaj mase ili drugih dinamičkih parametara može biti teško uočljiv iz same putanje. S druge strane, putanje s izraženim ubrzavanjem, usporavanjem ili promjenama smjera mogu sadržavati više informacija o dinamici vozila. U radu se zato naglašava da estimacija parametara ne daje uvijek jednako pouzdane rezultate, nego ovisi o vrsti promatrane putanje, kvaliteti ulaznih podataka i prikladnosti korištenog modela vozila. Unatoč tome, predloženi pristup pokazuje mogućnost povezivanja rekonstrukcije scenarija s fizičkim svojstvima vozila.

Za razvoj i procjenu predloženih metoda korišteni su stvarni i sintetički skupovi podataka. Od stvarnih skupova podataka korišteni su HighD i NGSIM. Ti skupovi sadrže putanje vozila dobivene obradom videozapisa prometa i predstavljaju vrijedan izvor stvarnih prometnih ponašanja. HighD sadrži putanje vozila snimljene iz zraka na njemačkim autocestama, dok NGSIM sadrži podatke snimljene na prometnicama u Sjedinjenim Američkim Državama. Takvi podaci omogućuju ispitivanje metoda na realnim putanjama, ali imaju i praktična ograničenja. Podaci mogu sadržavati šum, pogreške detekcije, nepravilnosti uzorkovanja i nesigurnosti u određivanju prometnih trakova. Osim toga, oznake početka i završetka promjene prometne trake nisu uvijek dostupne u obliku koji je potreban za procjenu logičkih segmenata. Zbog potrebe za podacima s poznatim referentnim oznakama u radu je korišten i sintetički skup HLCsim. Taj skup omogućuje kontrolirano ispitivanje metoda jer se za svaku putanju može znati koja se radnja izvodi i kada počinje i završava. Sintetički podaci posebno su važni za procjenu metoda klasifikacije i metrika, jer omogućuju usporedbu s poznatom referencom. U radu su takvi podaci korišteni za ispitivanje sposobnosti metoda da rekonstruiraju logičke radnje iz konkretnih putanja. Skup HLCsim obuhvaća 2912 scenarija vožnje, približno 1002 kilometra simulirane vožnje i više od 20 sati podataka, uključujući promjene prometne trake ulijevo i udesno te zadržavanje prometne trake. Podaci su dodatno prošireni i pripremljeni kako bi se povećala raznolikost primjera. Posebna pozornost posvećena je ispitivanju robusnosti metoda na šum i poremećaje. Snimljene putanje vozila u stvarnim skupovima podataka rijetko su idealne. Mogu sadržavati mjerni šum, izdvojene pogrešne vrijednosti, male oscilacije položaja i nepravilnosti koje nisu

posljedica stvarnog gibanja vozila. Kako bi se ispitalo ponašanje metoda u takvim uvjetima, podacima su dodavane različite razine šuma. Time se može procijeniti koliko klasifikacija ostaje stabilna kada kvaliteta podataka opada. Ova je analiza važna jer metode namijenjene stvarnoj uporabi moraju biti sposobne raditi s podacima različite kvalitete. Pokazano je da razina šuma može znatno utjecati na detekciju granica radnji i kontinuitet klasifikacije, zbog čega je u radu uveden i poseban postupak filtriranja putanja. Uz stvarne i sintetičke putanje za klasifikaciju, u radu je uveden i strukturirani sintetički referentni skup scenarija za provjeru cjelokupnog postupka rekonstrukcije. Takvi referentni scenariji omogućuju ispitivanje postupka pod kontroliranim uvjetima, jer je poznata izvorna logička struktura scenarija. Iz logičkog opisa može se generirati konkretna putanja, zatim se na nju mogu dodati poremećaji, a nakon toga se može provesti rekonstrukcija i usporediti dobiveni rezultat s poznatom referencom. Na taj se način može ocijeniti ne samo točnost pojedine klasifikacijske metode, nego i kvaliteta cijelog lanca obrade od putanje do logičkog scenarija. Za procjenu rezultata klasifikacije na način primjeren generiranju logičkih scenarija, u radu su predložene posebne metrike za klasifikaciju bočnih radnji vozila. Uobičajene metrike, poput ukupne točnosti, preciznosti, odziva i F-mjere, korisne su za opću procjenu klasifikacijskog modela, ali ne daju potpunu informaciju o kvaliteti rezultata za potrebe scenarija. Kod logičkih scenarija posebno su važne granice radnji i kontinuitet segmenata. Primjerice, klasifikator može ostvariti visoku ukupnu točnost, ali ako početak promjene trake detektira prekasno ili završetak prerano, rekonstruirani scenarij neće dobro opisivati stvarni manevar. Isto tako, kratki prekidi unutar jedne radnje mogu dovesti do nepotrebnog stvaranja više segmenata i otežati izradu logičkog opisa. Predložene metrike kvantificiraju točnost detekcije početka i završetka radnji, prostorno odstupanje detektiranih manevara u odnosu na referentne središnje linije te vremenski kontinuitet klasifikacije unutar pojedine radnje. Metrika početka i završetka ocjenjuje koliko su granice detektirane radnje udaljene od referentnih granica. Metrika prostornog odstupanja ocjenjuje položaj detektiranog manevra u odnosu na referentnu sredinu manevra. Metrika uporabljivosti procjene uzima u obzir može li se rezultat klasifikacije izravno pretvoriti u smislen logički segment ili sadrži prekide, dodatne detekcije i nestabilnosti koje bi zahtijevale naknadnu ručnu obradu. Takve metrike daju potpuniju sliku o kvaliteti rezultata jer su prilagođene konačnoj namjeni, odnosno generiranju logičkih scenarija. Ove metrike posebno su važne jer pokazuju da visoka točnost klasifikacije ne mora uvijek značiti visoku uporabljivost za scenarije. Ako se radnja detektira s pomakom, ako se unutar jedne stvarne promjene trake pojave kratki prekidi ili ako se pojave lažno pozitivni manevri, dobiveni scenarij može biti neprikladan za simulacijsku uporabu. Stoga vrednovanje mora uključivati i vremensku i prostornu kvalitetu segmenata. U tom smislu predložene metrike povezuju klasifikacijski rezultat s praktičnim zahtjevima rekonstrukcije logičkih scenarija.

U radu su razmatrane dvije metode za klasifikaciju bočnih radnji vozila. Prva metoda zasniva se na polinomskom modeliranju u krivolinijskom koordinatnom sustavu, pri čemu se bočno gibanje aproksimira polinomom trećeg stupnja. Krivolinijski koordinatni sustav prikladan je za opis gibanja vozila na cesti jer omogućuje odvajanje uzdužnog gibanja duž ceste od bočnog pomaka u odnosu na prometni trak ili referentnu liniju. U takvom koordinatnom sustavu promjena prometne trake očituje se kao sustavna promjena bočne koordinate. Time se olakšava prepoznavanje trenutaka u kojima vozilo napušta jedan

prometni trak i prelazi u drugi. Za razliku od globalnog koordinatnog sustava, u kojem oblik ceste i zakrivljenost mogu otežati analizu, krivolinijski opis omogućuje izravnije tumačenje ponašanja vozila u odnosu na prometnu infrastrukturu. Kod polinomske metode ulazna putanja i prometne linije promatraju se kao funkcije koje se mogu aproksimirati i analizirati. Bočno gibanje vozila u okolini promjene prometne trake opisuje se polinomom trećeg stupnja. Takav polinom omogućuje dovoljno fleksibilan opis glatkog bočnog pomaka vozila, a istodobno ostaje jednostavan za tumačenje. Na temelju oblika polinoma, njegovih sjecišta s linijama prometnih trakova te promjene bočnog položaja moguće je odrediti smjer i granice manevra. Početak i završetak promjene trake nisu uvijek izravno vidljivi iz samih diskretnih uzoraka, osobito ako je uzorkovanje rijetko ili ako se prelazak preko linije događa između dva uzorka. Polinomska aproksimacija omogućuje procjenu tih trenutaka s većom prostornom i vremenskom dosljednošću. Prednost polinomske metode je njezina objašnjivost. Rezultat klasifikacije može se povezati s geometrijom putanje i položajem vozila u odnosu na prometne trakove. Metoda ne zahtijeva proces treniranja, ne ovisi o velikom broju ručno označenih primjera i može se primijeniti na podatke za koje su dostupni položaji vozila i informacije o prometnim linijama. Takav pristup posebno je koristan kada je važno razumjeti zašto je određeni manevar detektiran i kada je potrebno izbjeći ovisnost o crnoj kutiji podatkovno vođenog modela. Ograničenje metode je njezina osjetljivost na kvalitetu ulaznih podataka. Ako su putanje jako narušene šumom, ako su prometne linije neprecizno određene ili ako manevar ne odgovara pretpostavljenom glatkom obliku, točnost detekcije može se smanjiti. Druga metoda koristi klasifikator zasnovan na neuronskim mrežama, koji predstavlja podatkovno vođen pristup prepoznavanju bočnih radnji vozila. Za razliku od polinomske metode, neuronska mreža ne koristi unaprijed zadano pravilo za određivanje manevra, nego iz označenih podataka uči obrasce koji odgovaraju pojedinim radnjama. Ulaz u takav klasifikator čine vremenski nizovi značajki izvedenih iz položaja vozila i odnosa prema prometnim trakovima. Arhitektura razmatrana u radu koristi konvolucijske slojeve za izdvajanje lokalnih značajki i dvosmjerne rekurentne slojeve za obradu vremenskog konteksta. Time se omogućuje da klasifikator prepozna ne samo trenutni položaj vozila, nego i razvoj gibanja prije i poslije promatranog trenutka. Neuronska mreža posebno je prikladna za rad s podacima koji sadrže šum, različite oblike manevara i promjenjive vremenske duljine radnji. Budući da uči iz primjera, može prepoznati obrasce koji nisu jednostavno opisani jednim analitičkim pravilom. U radu se takav pristup koristi kao druga, komplementarna metoda klasifikacije bočnih radnji vozila. Rezultati pokazuju da klasifikator zasnovan na neuronskim mrežama može postići vrlo dobru ukupnu točnost i robusnost, osobito pri većim razinama šuma. Međutim, takav pristup zahtijeva kvalitetno označene podatke za treniranje i provjeru. Također, rezultati mogu ovisiti o sličnosti između podataka korištenih za treniranje i podataka na koje se mreža kasnije primjenjuje. Zbog toga je u radu važno usporediti neuronsku mrežu s interpretabilnijom polinomskom metodom i procijeniti njihove prednosti i ograničenja. Usporedba dviju metoda provedena je s naglaskom na konačnu uporabljivost rezultata za generiranje logičkih scenarija. Uobičajena klasifikacijska točnost sama po sebi nije dovoljna. Kod pretvorbe putanje u scenarij nije važno samo koliko je pojedinačnih uzoraka ispravno označeno, nego i jesu li ispravno detektirani početak i završetak radnje, je li detekcija prostorno centrirana u odnosu na stvarni manevar i stvara li klasifikator kontinuirane segmente koji se mogu izravno upotrijebiti. Kratki prekidi u

klasifikaciji, lažno pozitivne detekcije ili pomaknute granice radnji mogu imati velik utjecaj na kvalitetu rekonstruiranog logičkog scenarija. Zbog toga se metode u radu ne vrednuju samo kao klasifikatori, nego kao dijelovi postupka za automatsku izgradnju scenarija. Osim klasifikacije bočnih radnji, u radu je razmatrana i klasifikacija uzdužnog gibanja vozila. Uzdužno gibanje opisuje promjene položaja, brzine i ubrzanja vozila u smjeru vožnje. Za potrebe logičkog scenarija nije nužno zadržati svaki pojedini uzorak iz snimljene putanje. Naprotiv, poželjno je dobiti jednostavniji opis koji zadržava bitna svojstva gibanja, ali se može lako parametrizirati i ponovno izvesti u simulaciji. U radu se uzdužno gibanje opisuje segmentima s približno stalnim ubrzanjem. Takvi segmenti omogućuju opis ubrzavanja, usporavanja i gibanja približno stalnom brzinom. Time se kontinuirana i često šumom narušena putanja pretvara u niz jednostavnijih uzdužnih radnji. Postupak određivanja uzdužnih segmenata temelji se na smanjenju broja čvorova putanje i aproksimaciji gibanja jednostavnijim dijelovima. Cilj nije savršeno reproducirati svaku lokalnu promjenu brzine, nego zadržati one promjene koje su važne za scenarijski opis. Prevelik broj segmenata doveo bi do scenarija koji je gotovo jednako složen kao izvorna putanja i teško bi se prilagođavao. Premalen broj segmenata mogao bi izgubiti važne značajke gibanja. Zato se traži ravnoteža između točnosti opisa i jednostavnosti logičkog scenarija. Uzdužni segmenti zatim se kombiniraju s bočnim radnjama kako bi se dobio potpun opis gibanja vozila.

Formiranje logičkih segmenata predstavlja središnji korak u pretvorbi putanje u scenarij. Svaki segment definiran je bočnom i uzdužnom radnjom te pripadajućim parametrima. Ako vozilo zadržava prometnu traku i giba se približno stalnom brzinom, taj dio putanje može se opisati jednim segmentom. Ako vozilo počne mijenjati prometnu traku, uvodi se novi segment. Ako se tijekom promjene trake istodobno promijeni uzdužno ubrzanje, granice segmenata moraju odražavati i tu promjenu. Na taj se način dobiva opis koji prati semantičke promjene u ponašanju vozila. Takav opis prikladniji je za logički scenarij od izravnog niza položaja jer omogućuje promjenu parametara unutar pojedinih radnji. Uz same segmente, potrebno je odrediti i početne uvjete scenarija. Početni uvjeti uključuju položaj vozila, brzinu, smjer gibanja, prometni trak, bočni odmak i druge veličine potrebne za izvođenje scenarija u simulacijskom okruženju. Ako se rekonstruirani scenarij želi ponovno izvesti, nije dovoljno znati samo redoslijed radnji. Potrebno je znati odakle vozilo polazi, kojom se brzinom kreće i u kakvom je odnosu prema prometnoj infrastrukturi. U radu se zato razmatra način izdvajanja tih veličina iz snimljene putanje i njihovo povezivanje s formiranim logičkim segmentima. Time se stvara cjelovit opis koji se može pretvoriti u konkretan simulacijski slučaj. Eksperimentalni dio rada obuhvaća ispitivanje pojedinih dijelova predloženog okvira i provjeru cjelokupnog postupka. Metode klasifikacije bočnih radnji ispitane su na pripremljenim skupovima podataka i pri različitim razinama šuma. Rezultati pokazuju da polinomska metoda može dati dobre i interpretabilne rezultate, osobito kada su ulazni podaci dovoljno kvalitetni. Neuronska mreža pokazuje veću robusnost u uvjetima jačeg šuma i postiže bolju ukupnu klasifikacijsku učinkovitost. Međutim, nove metrike pokazuju da se razlike među metodama ne smiju promatrati samo kroz ukupnu točnost. U nekim slučajevima metoda s nešto manjom ukupnom točnošću može dati prostorno bolje centrirane ili uporabljivije segmente, dok metoda s većom točnošću može stvarati kratke prekide ili dodatne radnje koje otežavaju rekonstrukciju scenarija.

Cjelokupni postupak pretvorbe putanja u logičke scenarije provjeren je na sintetičkim

referentnim scenarijima i na stvarnim putanjama. Sintetički scenariji omogućuju provjeru zato što je poznat izvorni logički opis. Iz takvog se opisa može generirati konkretna putanja, zatim se na nju mogu primijeniti metode klasifikacije i segmentacije, a dobiveni rezultat može se usporediti s poznatom referencom. Time se može procijeniti koliko dobro postupak rekonstruira izvorne radnje i granice segmenata. Primjena na stvarne putanje, posebno na podatke iz skupa HighD, pokazuje da se predloženi pristup može koristiti i na realnim prometnim situacijama. Pritom se pojavljuju ograničenja vezana uz kvalitetu stvarnih podataka, nesigurnost oznaka, geometriju prometnih trakova i razlike između stvarnog vozila i korištenog modela.

Rezultati rada pokazuju da automatizirana pretvorba snimljenih putanja u logičke scenarije zahtijeva usklađivanje više povezanih postupaka. Kvaliteta konačnog scenarija ne ovisi samo o klasifikatoru bočnih radnji, nego i o predobradi podataka, opisu uzdužnog gibanja, načinu formiranja segmenata, određivanju početnih uvjeta i odabiru metrika za procjenu. Ako je ulazna putanja jako poremećena, klasifikacija može biti nestabilna. Ako su granice bočnih ili uzdužnih radnji pogrešno određene, logički segmenti neće vjerno opisivati stvarno ponašanje vozila. Ako se koriste samo opće metrike klasifikacije, može se pogrešno zaključiti da je metoda prikladna za scenarije iako rezultat zahtijeva znatnu naknadnu obradu. Zbog toga je doprinos rada u cjelovitom promatranju problema, od sirovih putanja do logičkih segmenata. Predloženi okvir ima više doprinosa. Prvi doprinos je metoda klasifikacije bočnih radnji vozila zasnovana na polinomskom modeliranju u krivolinijskom koordinatnom sustavu. Drugi doprinos je primjena klasifikatora zasnovanog na neuronskim mrežama za isti problem i usporedba njegovih svojstava s interpretabilnim polinomskim pristupom. Treći doprinos je postupak formiranja logičkih segmenata kombiniranjem bočnih i uzdužnih radnji vozila. Četvrti doprinos je simulacijski postupak filtriranja putanja i estimacije parametara vozila zasnovan na proširenoj estimaciji pokretnog horizonta i principima upravljanja zasnovanog na predikciji modela. Peti doprinos su metrike za procjenu klasifikacije bočnih radnji prilagođene generiranju logičkih scenarija. Zajedno, ti doprinosi čine cjelinu usmjerenu na automatiziranu rekonstrukciju scenarija iz podataka. Rad također pokazuje važnost prikladnih skupova podataka za razvoj i evaluaciju metoda. Stvarni skupovi podataka potrebni su jer sadrže stvarna prometna ponašanja i neidealnosti mjerenja. Sintetički skupovi potrebni su jer omogućuju poznatu referencu i kontrolirano ispitivanje. Kombiniranjem tih izvora moguće je dobiti potpuniju sliku o ponašanju metoda. Sintetički podaci omogućuju preciznu usporedbu s referentnim oznakama, dok stvarni podaci pokazuju kako se metode ponašaju u praktičnim uvjetima. U radu se zato ne oslanja samo na jednu vrstu podataka, nego se nastoji povezati kontrolirana evaluacija s provjerom na stvarnim putanjama. Predloženi pristup ima i ograničenja koja su važna za tumačenje rezultata. Točnost klasifikacije i rekonstrukcije ovisi o kvaliteti ulaznih putanja, dostupnosti i točnosti opisa prometnih trakova, razini šuma, obliku manevara i kvaliteti referentnih oznaka. Kod stvarnih skupova podataka referentne oznake često nisu idealne, pa se i evaluacija mora promatrati uz određenu nesigurnost. Nadalje, model vozila korišten u filtriranju i estimaciji parametara ne može potpuno opisati sva moguća stvarna vozila i sve uvjete vožnje. Procjena parametara ovisi o informativnosti putanje, a filtriranje zahtijeva dodatno računalno vrijeme. Ipak, budući da je cilj obrada podataka za naknadno generiranje scenarija, takva računalna složenost može biti prihvatljiva. Mogući

budući razvoj uključuje proširenje metoda na složenije prometne situacije i veći broj sudionika. Ovaj rad usredotočen je ponajprije na izdvajanje radnji pojedinačnog vozila iz njegove putanje, osobito u kontekstu autocestovne vožnje i promjena prometnih trakova. U stvarnim scenarijima važne su i interakcije među vozilima, odnos prema vozilima ispred i iza promatranog vozila, reakcije na prometnu infrastrukturu, uključivanje i isključivanje s ceste te složeniji gradski manevri. Logički scenariji koji uključuju više sudionika zahtijevaju dodatne metode za prepoznavanje međusobnih odnosa i uzročno-posljedičnih veza između radnji. Predloženi okvir može poslužiti kao osnova za takva proširenja jer već omogućuje izdvajanje osnovnih radnji i njihovo pretvaranje u segmentirani opis.

Zaključno, ovaj rad predlaže cjelovit okvir za pretvorbu snimljenih putanja vozila u logičke scenarije prikladne za simulacijska testiranja zasnovana na scenarijima. Okvir obuhvaća klasifikaciju bočnih radnji vozila, opis uzdužnog gibanja, formiranje logičkih segmenata, određivanje početnih uvjeta scenarija, filtriranje i fizički konzistentnu rekonstrukciju putanja, estimaciju parametara vozila te metrike prilagođene procjeni uporabljivosti rezultata za generiranje scenarija. Time se povećava ispitna vrijednost postojećih skupova snimljenih podataka jer se konkretne putanje ne koriste samo za ponovno izvođenje jedne zabilježene situacije, nego kao osnova za stvaranje prilagodljivih i ponovljivih logičkih scenarija. Zajedno, predložene metode klasifikacije, postupak predobrade i metrike omogućuju pouzdanu i skalabilnu pretvorbu snimljenih podataka vožnje u logičke scenarije, smanjujući ručni rad i povećavajući ponovljivost simulacijskih testiranja zasnovanih na scenarijima. Rezultati pokazuju da se iz snimljenih putanja mogu izdvojiti radnje vozila, da se te radnje mogu povezati u logičke segmente i da se kvaliteta takve pretvorbe može procjenjivati metrikama koje su prilagođene konačnoj primjeni. Time se doprinosi razvoju metoda za automatizirano generiranje scenarija, boljem iskorištavanju stvarnih prometnih podataka i učinkovitijem testiranju naprednih funkcija vozila u simulacijskom okruženju. Predloženi pristup ne zamjenjuje stručnu provjeru scenarija, ali znatno smanjuje količinu ručnog rada i daje metodološki temelj za daljnji razvoj automatizirane rekonstrukcije scenarija vožnje iz podataka.

KLJUČNE RIJEČI: putanja vozila, logički segmenti, testiranje zasnovano na scenarijima, klasifikacija promjene prometne trake, Polinomisko modeliranje, neuronske mreže, estimacija pokretnog horizonta, promjena prometne trake, simulacija segmenata, simulacija vožnje vozila, detekcija, detekcija scenarija

CONTENTS

1 INTRODUCTION 1
1.1 Motivation and Problem Statement 1
1.2 Original Scientific Contributions 2
1.3 Outline of the Thesis 3

2 GENERAL BACKGROUND 5
2.1 Introduction 5
2.2 Scenario-based Testing 6
2.2.1 Data-driven Scenarios 6
2.2.2 Related Work 7
2.3 Curvilinear Coordinate System 7
2.4 Polynomial and Spline-Based Modelling 8
2.4.1 Polynomial Modelling 8
2.4.2 Greedy Knot Reduction 9
2.5 Machine Learning 10
2.5.1 Foundations of Machine Learning 10
2.5.2 Neural Network Architectures 11
2.6 Vehicle Modelling and Model Predictive Control 12
2.6.1 Vehicle Modelling 12
2.6.2 Model Predictive Control 15
2.7 Artificial Disturbances 17
2.7.1 Noise 17
2.7.2 Outliers 17
2.7.3 Irregular Sampling 17
2.8 Rauch Tung Striebel Smoother 18
2.9 Summary 19

3 EXTENDED HORIZON MODEL-PREDICTIVE TRAJECTORY ESTIMATION AND PARAMETER ESTIMATION 20
3.1 Introduction 20
3.2 Patching and Reconstruction of Vehicle States 21
3.2.1 MPC Modification for Filtering 22
3.2.2 Evaluation 24
3.3 Vehicle Parameter Estimation 27
3.3.1 Evaluation on HighD Trajectories 28

3.3.2 Evaluation on Logic Scenario Simulation . . . 31
3.4 Summary . . . 38

4 DATASETS AND NOVEL METRICS . . . 39
4.1 Introduction . . . 39
4.2 Datasets for Lane Change Classification . . . 39
4.2.1 HighD . . . 39
4.2.2 NGSIM . . . 40
4.2.3 HLCsim . . . 40
4.2.4 Dataset Composition and Augmentation . . . 41
4.2.5 Synthetic Reference Dataset . . . 41
4.3 Novel Metrics . . . 43
4.3.1 Centre Offset . . . 44
4.3.2 Start/End Metric . . . 45
4.3.3 Estimate Usability . . . 45
4.4 Summary . . . 47

5 VEHICLE ACTION CLASSIFICATION AND LOGIC SEGMENTS . . . 48
5.1 Introduction . . . 48
5.2 Classification of Lateral Vehicle Actions . . . 49
5.2.1 Polynomial Lane Change Classification Method . . . 50
5.2.2 Neural Network Lane Change Classification Method . . . 54
5.3 Classification of Longitudinal Vehicle Actions . . . 65
5.3.1 Greedy Knot Constant-Acceleration Segments . . . 65
5.4 Logic Segments Formation . . . 68
5.4.1 Initial Conditions . . . 68
5.4.2 Segment Forming . . . 69
5.5 Evaluation at the Scenario Level . . . 69
5.5.1 Reference Logical Scenarios . . . 70
5.5.2 Evaluation with the HighD Dataset . . . 99
5.6 Summary . . . 103

6 CONCLUSION AND OUTLOOK . . . 104

BIBLIOGRAPHY . . . 106

CURRICULUM VITAE . . . 114
6.0.1 Journal Publications: . . . 115

ŽIVOTOPIS . . . 116

1

Introduction

The complexity of modern road vehicles is continually increasing, driven by both legislative requirements and user expectations for improved safety, comfort, usability, connectivity, efficiency, and other features. In particular, the demand for advanced systems such as autonomous driving and driver assistance technologies further contributes to this complexity. The development of such vehicles has, therefore, become increasingly challenging, requiring large volumes of data and extensive testing. To reduce testing time, improve experimental repeatability, and lower costs, a significant portion of testing and validation is performed in simulation environments. For these simulations to be effective and representative, they must approximate real-world driving conditions as closely as possible. To encompass all elements of such simulations, the automotive industry uses the term scenario-based simulation testing. Scenario-based simulation testing provides a structured, repeatable, and adaptive framework for evaluating the behaviour of a vehicle and its systems, thereby, enabling the safe and cost-effective development of solutions in the automotive domain. Driving scenarios can be divided into two types: *non-adaptive* and *adaptive*. In non-adaptive test scenarios, the movements of all dynamic objects and their behaviours are fully predefined. Adaptive test scenarios do not predetermine the movements of dynamic objects such as other vehicles; instead, these objects are modelled to react to changes in the environment and to interactions with each other. Vehicle scenarios are typically classified into three abstraction levels: *concrete*, *logical*, and *functional* A functional scenario has the highest level of abstraction and is essentially just an idea for a driving situation. Concrete (*replay*) scenarios represent exact replays of recorded trajectories, with no possibility of easily altering parameters. Logic scenarios provide a higher level of abstraction by describing vehicle actions with logical segments, which allows flexible parameter variation and enables scalable testing and validation.

1.1 Motivation and Problem Statement

Logic scenarios are mostly created manually and have little or no direct relation to real-world driving data. However, scenarios based on real-world data are more desirable for testing and validation. While generating replay scenarios from recorded data is common practice, the creation of logic scenarios from recorded data is rarely performed, as it requires substantial manual effort. Manual conversion from replay to logic scenarios is feasible but inefficient, and it becomes impractical when dealing with large datasets typically required

for testing and validation in the automotive industry. The motivation of this research is to define a reliable approach for automating such a conversion. This would enable the efficient generation of large numbers of logical scenarios and provide test cases that are more closely aligned with real-world driving conditions.

This thesis addresses the challenge of representing vehicle trajectories using logic segments that can be applied across multiple widely used simulation platforms. In other words, it seeks to increase the level of abstraction by transforming replay scenarios derived from recorded data into logic scenarios suitable for large-scale simulation and testing. The crucial difficulty in this process lies in the reliable classification of vehicle actions, so that these actions can be systematically represented using predefined logic segment types.

1.2 ORIGINAL SCIENTIFIC CONTRIBUTIONS

The contributions of this thesis aim to advance automated scenario generation by providing new methods that increase the abstraction level of recorded vehicle trajectories and, thereby, extend their testing value in simulation environments. The main scientific contributions are as follows:

- Polynomial method for lateral lane change classification. This thesis introduces a polynomial-based approach for detecting and classifying lateral vehicle manoeuvres.

  Vehicle trajectories are represented in the curvilinear coordinate system and approximated with a third-order polynomial, whose inflection points correspond to the start and end of lane change actions. This approach allows lane keeping and lane change manoeuvres to be distinguished without the need for training data or large labelled datasets, which are often scarce or expensive to obtain. Compared to machine learning models, the polynomial method is dataset-agnostic, lightweight, and provides interpretable parameters that can be directly mapped to logic scenario descriptions. The method was validated on publicly available datasets as well as a newly generated simulation dataset, demonstrating competitive accuracy while offering advantages in transparency and adaptability.

- Novel metrics for lateral lane change classification. Standard metrics such as accuracy and confusion matrices are insufficient for assessing the usefulness of classification results in the context of scenario generation, since they only capture overall correctness and fail to account for temporal or spatial aspects of vehicle manoeuvres.

  To address this gap, this thesis proposes new evaluation metrics specifically tailored for lane change classification. These metrics evaluate: (i) the accuracy of detecting the start and end points of lane changes, (ii) the positional offset of the detected manoeuvre relative to the ground truth, and (iii) the temporal continuity of the classification within each manoeuvre. By considering these aspects, the metrics provide a more detailed and practically relevant assessment of classification quality, enabling fair comparison between different methods and better alignment with the requirements of logic scenario generation.

- Simulation-based filtering method. A filtering method for recorded vehicle trajectories is developed based on moving horizon estimation principles and simulation consistency.

  Unlike conventional smoothing techniques, which often suppress relevant manoeuvre details or introduce unrealistic artifacts, the proposed method evaluates the plausibility of trajectories with respect to a vehicle dynamics model used in simulation. By constraining the filtering process to ensure physically realistic longitudinal and lateral motion, the method effectively eliminates measurement noise, sensor disturbances, and other irregularities that are common in real-world data. This ensures that the resulting trajectories are not only smoother but also simulation-ready, providing a robust foundation for subsequent classification and logic scenario generation.

- Synthetic dataset for highway driving with ground truth lane change annotations. To support training, evaluation, and benchmarking of classification methods, this thesis introduces a synthetic dataset of highway driving scenarios containing ground truth labels for lane change actions.

  The dataset was generated using a high-fidelity simulation environment, ensuring realistic vehicle dynamics and traffic interactions. Each trajectory is annotated with precise start and end points of lane changes, along with lane indices, enabling systematic validation of classification algorithms. With 2912 unique trajectories covering diverse lane configurations and manoeuvres, this dataset fills a critical gap, as no comparable public dataset with lane change ground truth exists to date. It, therefore, provides an essential foundation for reproducible research and fair comparison of lane change detection methods.

1.3 Outline of the Thesis

This thesis is structured into five main chapters, followed by the conclusion. The opening chapter introduces the research motivation, problem statement, and the original scientific contributions. An overview of the remaining chapters is provided below:

Chapter 2 The background and terminology required for the remainder of the thesis are established in this chapter. It covers scenario-based testing and standard abstraction levels (functional, logic, and concrete), with an emphasis on data-driven logic scenarios and automated conversion from recorded trajectories. Related work in data-driven scenario generation is reviewed, and the thesis contributions are placed in context. The chapter also introduces the technical foundations used later, including the curvilinear coordinate system, polynomial and spline-based modelling (with greedy knot reduction), core machine learning concepts for manoeuvre recognition, and control-oriented vehicle modelling with model predictive control (MPC). Artificial disturbances used in the experiments to model noise, outliers, and irregular sampling are also defined.

Chapter 3 An extended-horizon MPC-based method for vehicle trajectory filtering and reconstruction is developed in this chapter. The approach reconstructs trajectories by driving a simulated vehicle model to track measured data, yielding disturbance-reduced, high-resolution trajectories with complete state and control histories. Preprocessing steps for patching missing samples and reconstructing vehicle states are described, followed by the modified MPC formulation with future-state weighting. The method is evaluated on distorted Single lane change (SLC) / Duoble lane change (DLC) trajectories and compared with an RTS smoother. The same framework is further applied to vehicle parameter estimation, with a focus on vehicle mass estimation using both HighD trajectories and simulated logic scenarios.

Chapter 4 This chapter presents the datasets used in this thesis, detailing their acquisition, labelling, and augmentation for vehicle lateral action (lane-change) classification. It combines two real-world trajectory datasets (*HighD* and *NGSIM*) with a custom simulation dataset with ground-truth labels (*HLCsim*), and introduces a structured synthetic reference scenario dataset (*LC Scenarios for Lane Change Classification*) for validating scenario creation workflows under systematic parameter variations and controlled distortions. In addition, evaluation metrics specifically tailored to lane-change classification are defined to support logic scenario reconstruction. Limitations of standard measures such as accuracy and confusion matrices are discussed, particularly with respect to segmentation continuity, alignment, and action timing. Three dedicated metrics—*Centre Offset*, *Start/End*, and *Estimate Usability*—are introduced and illustrated through representative examples, figures, and tables to better capture manoeuvre alignment, temporal consistency, and the impact of false detections on practical scenario usability.

Chapter 5 The methodology for vehicle action classification and its conversion into logic scenarios is detailed in this chapter. Lateral actions are classified using both polynomial modelling and neural network approaches, while longitudinal actions are identified through greedy knot reduction into constant-acceleration segments. The integration of lateral and longitudinal classifications into logic segments is described, followed by the construction of complete logic scenarios. The chapter concludes with an end-to-end evaluation of the full process using both synthetic data and publicly available trajectory datasets.

Chapter 6 The main findings of the thesis are summarized in this chapter, and directions for future research are outlined.

2

General Background

This chapter introduces scenario-based testing and the different levels of scenario abstraction. It further provides an overview of related work and establishes the context for the remainder of this thesis. In addition, the chapter presents an introduction to the curvilinear coordinate system, polynomial and spline modelling, machine learning methods, vehicle dynamics modelling, and MPC.

2.1 Introduction

This chapter provides the background concepts and technical foundations required for the remainder of the thesis. It establishes the context of scenario-based testing as the primary paradigm for validation of advanced driver assistance and automated driving systems, and introduces the standard abstraction levels used in simulation-based evaluation. Special emphasis is placed on data-driven logic scenarios, as they offer a scalable compromise between realism and flexibility, while also exposing the need for automated conversion methods from recorded trajectories. Beyond the scenario framework, the chapter consolidates the key modelling and algorithmic tools that are later employed for vehicle action classification, trajectory reconstruction, and filtering. These include the curvilinear coordinate system for road-aligned trajectory representation, polynomial and spline-based approximation methods for trajectory modelling, machine learning techniques for manoeuvre recognition, and control-oriented vehicle models suitable for predictive control. The chapter further introduces MPC as a central optimisation-based framework used later for trajectory filtering and parameter estimation. To ensure reproducible and controlled evaluation, common disturbance types encountered in real-world trajectory data are formally defined, and the Rauch–Tung–Striebel (RTS) smoother is presented as a reference baseline for offline state estimation.

By consolidating these concepts into a unified background, this chapter provides the theoretical and practical basis for the subsequent chapters, where automated vehicle action classification, logic scenario generation, and MPC-based filtering methods are developed and evaluated.

2.2 SCENARIO-BASED TESTING

Scenario-based testing emerged in response to the impracticality of validating complex vehicle systems solely through mileage-based road testing. Early studies demonstrated that safety-critical events would require prohibitively long driving distances to achieve statistical confidence. Kalra and Paddock [1] further emphasized this challenge, estimating that real-world testing of certain advanced driver assistance systems (ADAS) functions could take up to 500 years and require more than 11 billion miles of driving. This motivation was also reflected in early international standardisation efforts, where International Organization for Standardization (ISO) introduced scenario-oriented test procedures for adaptive cruise control, forward collision warning, and lane departure warning systems [2, 3, 4]. Ulbrich et al. [5] subsequently defined and clarified the terminology of *scene*, *situation*, and *scenario* in the automotive context. Building on this foundation, Menzel et al. [6] formally introduced the classification of scenarios into three categories: *functional*, *logic*, and *concrete*, which has since become the standard framework for simulation-based testing. A *functional scenario* represents the highest level of abstraction and is essentially a description of a driving idea or situation. For example, a vehicle driving on a highway overtakes another vehicle while accelerating. At the opposite end, *concrete scenarios*, often referred to as *replay scenarios*, are exact replays of recorded trajectories with no possibility of easily altering parameters. Between these two extremes, *logic scenarios* provide a structured yet flexible representation of driving behaviour, described through segments such as lane keeping, lane changing, braking, and accelerating. This level of abstraction enables scalable testing and validation by allowing systematic variation of scenario parameters.

2.2.1 *Data-driven Scenarios*

There are two possible paths for defining a scenario for simulation-based testing. One is a manual approach, based on prior knowledge and experience, and the other is a *data-driven* approach, where recordings of real-world driving are converted into scenarios. Data-driven scenarios provide significantly greater value for testing, as they are more representative of real-world driving conditions. Creating data-driven concrete scenarios is a straightforward and widely used process. A vehicle trajectory, together with information about the road, is processed into a replayable scenario and executed in simulators. The most valuable, however, are data-driven logic scenarios, as they allow systematic variation of scenario parameters. By manipulating parameters from a single recorded scenario, multiple new test cases can be derived, and critical cases can be tested more effectively. Data-driven logic scenarios are typically created by manually interpreting vehicle movements from recorded trajectories or concrete scenarios, relying heavily on the expertise of skilled engineers. However, this process is time-consuming and does not scale well to large datasets. Therefore, there is a clear need for automated methods that can systematically convert concrete scenarios into logic representations.

The main challenge in automated conversion of data-driven scenarios lies in classifying vehicle actions in input trajectories into segments that can be mapped to logic scenario elements in the target simulation tool. Classification of vehicle actions is covered in chapter 5. Vehicle actions are typically divided into two groups: lateral and longitudinal. The primary

focus of this thesis is the classification of lateral actions. To address this, a novel polynomial method is introduced in Subsection 5.2.1. A comparison with the polynomial method is provided using a neural network approach, described in Subsection 5.2.2. Both methods, along with their comparative performance, were previously presented by the author in [7], and are further detailed and extended in Chapter 5. Finally, Section 5.3 addresses the classification of longitudinal vehicle actions, implemented using a grid-based knot reduction method.

2.2.2 *Related Work*

Early work by Ulbrich et al. [5] and Menzel et al. [6] established the conceptual foundation for scenario-based testing, providing terminology and categorization that later enabled systematic data-driven approaches. Building on this foundation, Zhao et al. [8] proposed converting real-world driving data into simulation-ready scenarios using ontology-based methods [9], highlighting the importance of structuring raw data into reusable scenario formats. Around the same time, the authors of [10] emphasized the alignment of data-driven scenarios with functional safety standards, presenting an approach for generating simulation scenarios within the framework of ISO 26262 [11]. Cai et al. [12] provided a comprehensive survey, categorising methods for parameterisation, criticality assessment, and representativeness of data-driven scenarios. Generative approaches have also emerged, including probabilistic models for constructing roundabout scenarios from trajectory data [13] and the *DriveSceneGen* framework [14], which synthesizes diverse simulation cases directly from large-scale driving datasets. More recently, Nguyen et al. [15] proposed *AVASTRA*, a reinforcement learning–based framework that combines vehicle states with environmental factors to generate safety-critical scenarios. At the same time, the CommonRoad *Scenario Factory 2.0* [16] integrated these advances into a modular framework for scenario generation, labelling, and parameter variation, offering a scalable toolchain for the creation of reproducible, data-driven scenarios.

Together, these contributions illustrate the evolution of data-driven scenario generation from ontology-based and standards-driven approaches to modern generative and reinforcement learning–based frameworks, reflecting the increasing importance of realism, variability, and scalability in simulation-based testing.

2.3 Curvilinear Coordinate System

In vehicle trajectory modelling, the curvilinear coordinate system (CCS) provides a road-aligned representation of motion relative to a reference path. Unlike Cartesian coordinates (x, y), CCS describes motion through the longitudinal displacement s along the path and the lateral deviation d from it [17, 18], as illustrated in Fig. 2.1. Conversion from global to curvilinear coordinates is performed by projecting the vehicle global position (x, y) onto the reference curve. Road lane lines can be represented in the same way, as arrays of lateral offsets d with respect to s, which enables consistent description of both the vehicle trajectory and lane boundaries. This representation is particularly advantageous in structured environments such as highways, where it simplifies manoeuvre description,

facilitates lane-change detection, and allows straightforward manipulation of trajectory data in both position and time domains.

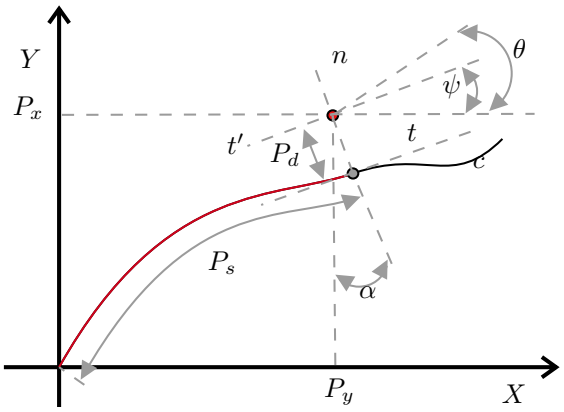


Figure 2.1: Curvilinear coordinate system compared to Cartesian coordinates.

For a vehicle located at $\mathbf{p} = (x, y)$ in Cartesian coordinates, the corresponding curvilinear coordinates are obtained by projecting $\mathbf{p}$ onto the reference path γ, parameterised by arc length s:

$$s^* = \arg\min_s \|\mathbf{p} - \gamma(s)\|, \tag{2.1}$$

$$d = \big(\mathbf{p} - \gamma(s^*)\big) \cdot \mathbf{n}(s^*). \tag{2.2}$$

Here, $\mathbf{t}(s)$ and $\mathbf{n}(s)$ denote the tangent and normal vectors of $\gamma(s)$, respectively. The coordinate s^* represents the closest point on the reference path, while d gives the signed lateral offset from it. The vehicle orientation ψ can be expressed relative to the road heading $\theta(s^*)$:

$$\Delta\psi = \psi - \theta(s^*). \tag{2.3}$$

This coordinate system decouples longitudinal and lateral dynamics, simplifying trajectory tracking, manoeuvre classification, and scenario abstraction [19, 20]. Lane keeping corresponds to small deviations in d, while lane changes manifest as structured variations in $d(s)$. Although CCS requires robust projection algorithms to handle curved roads, intersections, and noisy trajectory data [21], it remains a standard representation in automated driving research due to its compactness and direct alignment with road geometry.

2.4 Polynomial and Spline-Based Modelling

2.4.1 *Polynomial Modelling*

One of the methods used to address the classification problem described in Section 1.1 is polynomial fitting. Polynomial fitting is the process of finding a polynomial function that best approximates a given set of data points by minimising the error between the polynomial curve and the data. Polynomial functions are among the most widely used mathematical tools for trajectory generation and approximation in vehicle dynamics and control. Their appeal lies in their simplicity, differentiability, and ability to represent smooth curves with relatively few parameters. A general polynomial of degree n can be expressed as:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n, \tag{2.4}$$

where x is the independent variable, typically representing time or arc length, and a_i are the polynomial coefficients. By selecting an appropriate degree and boundary conditions, polynomials can capture a wide variety of vehicle trajectories while ensuring continuity in position, velocity, and acceleration. The primary advantages of polynomial modelling are its closed-form representation, low computational cost, and straightforward differentiation, which facilitates the computation of velocity, acceleration, and curvature. However, polynomial trajectories also suffer from limitations. High-degree polynomials may lead to oscillatory behaviour (Runge's phenomenon), while low-degree polynomials may fail to capture complex manoeuvres. Piecewise polynomials or spline-based methods mitigate these issues by dividing trajectories into smaller segments, each approximated by a lower-degree polynomial [22]. Another challenge lies in robustness to noisy or incomplete data: fitting polynomials to disturbed trajectories may introduce artefacts unless combined with smoothing or regularisation. Some authors, such as Lai et al. [23] and Zhang et al. [24], have used polynomials for lane-change and vehicle trajectory planning. Polynomials are also used for trajectory generation by Werling et al. [18].

Polynomial modelling, implemented in this thesis primarily through `polyfit`-based least-squares fitting, provides a powerful tool for vehicle trajectory approximation and reconstruction. Its use enables both the detection of lane change actions through polynomial inflection points and the generation of logic scenario representations in curvilinear coordinates, bridging the gap between raw trajectory data and simulation-ready logic scenarios.

While polynomial fitting provides a global approximation, spline-based methods and Greedy Knot Reduction enable more flexible, piecewise representations with controlled complexity.

2.4.2 *Greedy Knot Reduction*

The problem of knot selection has a rich history in approximation theory and statistics. Foundational work on splines was introduced by de Boor et al. [25, 26], followed by Tiller [27], who proposed algorithms for systematic knot removal in spline representations. Further contributions were made by Ruppert [28] and Hansen and Kooperberg [29], who studied knot selection in the context of regression splines. Lee [30] provided a comparative study of regression spline algorithms, highlighting trade-offs between computational efficiency and approximation fidelity. Formally, given data points $\{(x_i, y_i)\}_{i=1}^{N}$, the goal is to approximate them by a spline function

$$S(x) = \sum_{j=1}^{K} c_j B_j(x), \tag{2.5}$$

where $B_j(x)$ are B-spline basis functions associated with a knot sequence $\mathcal{T} = \{t_0, t_1, \ldots, t_K\}$ and coefficients c_j. Greedy Knot Reduction operates by iteratively removing knots from an initial dense knot sequence. At each step, the algorithm selects the knot whose removal produces the smallest increase in approximation error, typically measured by the squared residual norm:

$$E = \sum_{i=1}^{N} \left(y_i - S(x_i)\right)^2 . \tag{2.6}$$

If $\mathcal{T}^{(k)}$ denotes the knot sequence at iteration k, then the update step is

$$\mathcal{T}^{(k+1)} = \mathcal{T}^{(k)} \setminus \{t_j^*\}, \tag{2.7}$$

where

$$t_j^* = \arg\min_{t_j \in \mathcal{T}^{(k)}} \Delta E(t_j), \tag{2.8}$$

and $\Delta E(t_j)$ is the increase in error when knot t_j is removed. This greedy process continues until a desired number of knots remains or the approximation error exceeds a predefined tolerance ε:

$$E \leq \varepsilon. \tag{2.9}$$

Thus, Greedy Knot Reduction provides a principled balance between model complexity (number of knots) and approximation fidelity.

2.5 MACHINE LEARNING

2.5.1 *Foundations of Machine Learning*

Machine learning (ML) is a subfield of artificial intelligence (AI) that focuses on developing algorithms that enable systems to learn patterns from data and improve performance without being explicitly programmed. It has emerged as one of the most influential fields in computer science, statistics, and engineering, driving progress by allowing systems to adapt and improve based on data rather than fixed programming [31, 32]. The origins of ML can be traced back to the 1950s, when early pioneers such as Alan Turing proposed the idea of machines that could learn from experience [33]. Arthur Samuel later introduced one of the first practical implementations of ML by developing a checkers-playing program that improved its performance through self-play [34]. In the following decades, foundational models such as Rosenblatt's perceptron [35] laid the groundwork for neural networks, while statistical approaches and decision-theoretic perspectives shaped the broader ML field [36]. Over time, the combination of theoretical advances, increased computational power, and the availability of large datasets led to the modern era of ML, where it has become one of the most influential areas in computer science, statistics, and engineering [31, 32].

ML methods are categorised into supervised learning, where models learn mappings from input data to labelled outputs; unsupervised learning, which finds hidden structures without labels; and reinforcement learning, where agents learn by interacting with environments to maximise rewards [37]. Common unsupervised algorithms are clustering, dimensionality reduction, and association rules. These are often used for pattern discovery, anomaly detection, and feature extraction [38]. Reinforcement learning algorithms can be divided into two categories: model-based methods and model-free methods. Model-based methods use or learn a model of the environment and are often applied in planning and robotics. Model-free methods, such as Q-learning and policy gradients, scale better and have achieved success in games and autonomous control [39, 40]. Common supervised algorithms are linear regression, logistic regression, decision trees, support vector machines (SVM), Naive Bayes, and neural networks. These are widely used for regression, classification, and prediction tasks [31, 32, 41].

Inspired by biological neural networks, commonly known as the brain, which consist of interconnected neurons, artificial neural networks are composed of multiple layers of perceptrons, a mathematical model of the neuron. The perceptron is a single decision-making unit that computes a weighted sum of its inputs, applies an activation function, and produces a binary output. A single perceptron is capable of classifying linearly separable data by finding a hyperplane that separates two classes [35, 36]. Initially, the development of neural networks was slow because the computational power required to optimise their parameters was not widely available. The resurgence of neural networks in the late 2000s, driven by increased computational power, large datasets, and optimisation techniques such as stochastic gradient descent and backpropagation, initiated the deep learning revolution [42].

2.5.2 *Neural Network Architectures*

Fully connected (FC) neural networks, also known as multilayer perceptrons (MLPs), were among the first practical architectures applied to supervised learning tasks. They consist of layers where every neuron is connected to all neurons in the preceding layer, enabling universal approximation of nonlinear functions [43]. Although powerful, FC networks treat all inputs equivalently and fail to exploit structured patterns such as spatial locality or temporal dependencies, limiting their efficiency for high-dimensional or sequential data. Convolutional neural networks (CNNs) represent one of the most impactful innovations in deep learning. By introducing local receptive fields, parameter sharing, and pooling layers, CNNs are able to learn hierarchical feature representations while significantly reducing the number of parameters [44]. CNNs excel in extracting spatially invariant patterns and are widely applied in image recognition, object detection, and semantic segmentation. Beyond images, CNNs have also been adapted to one-dimensional signals and multivariate time series, including vehicle trajectory sequences, where they capture local correlations between successive samples and road geometry. While CNNs are highly effective in capturing spatial dependencies, they lack mechanisms for modelling long-term temporal relationships. Recurrent neural networks (RNNs) were developed to address this limitation by introducing feedback connections that allow information to persist over time. However, standard RNNs suffer from vanishing and exploding gradient problems, limiting their ability to learn long sequences [45]. To overcome this, long short-term memory (LSTM) networks were introduced, incorporating gated memory cells that regulate the flow of information [46]. LSTMs are capable of capturing dependencies over long horizons, making them highly effective for sequential data such as speech, text, and vehicle trajectories. Despite their advantages, standard LSTMs only model information in a single temporal direction. This is a limitation for tasks where both past and future context are informative. Bidirectional LSTMs (Bi-LSTMs) extend the architecture by processing sequences in both forward and backward directions, yielding richer temporal representations [47]. Bi-LSTMs have been shown to improve performance in numerous sequence modelling tasks, including trajectory classification, where the evolution of manoeuvres can be better understood by jointly considering previous and future states [48, 49]. In the automotive domain, Bi-LSTMs enable more accurate recognition of manoeuvres such as lane changes, where bidirectional dependencies strongly

influence classification outcomes. Recent work has demonstrated that hybrid architectures combining CNNs and LSTMs further improve performance by leveraging complementary strengths. CNN layers capture local spatial or short-term temporal patterns, while LSTM or Bi-LSTM layers capture long-term dependencies [48]. Multi-scale CNNs (MSCNNs), which operate with filters of varying receptive field sizes, have also been integrated with Bi-LSTMs, allowing simultaneous modelling of local and global features. Such Bi-LSTM + MSCNN models have achieved state-of-the-art results in manoeuvre recognition. This type of architecture is adopted in this thesis for lateral vehicle action classification, as described in Section 5.2.

In summary, ML and neural networks provide a powerful framework for spatio-temporal modelling of vehicle trajectories. From classical FC networks to advanced Bi-LSTM + CNN hybrids, these models progressively address the challenges of capturing nonlinear, sequential, and noisy driving data. Their role in manoeuvre recognition directly supports the automated generation of logic scenarios from recorded trajectories, forming an essential step in scenario-based testing pipelines. As neural network architectures continue to evolve, their integration into automotive applications will play a central role in improving the safety, reliability, and efficiency of autonomous driving systems.

2.6 Vehicle Modelling and Model Predictive Control

This section presents the mathematical modelling of the vehicle and the formulation of MPC applied in this thesis.

2.6.1 *Vehicle Modelling*

Mathematical modelling of vehicles provides the foundation for analysing, simulating, and controlling vehicle dynamics. By translating the physical behaviour of a vehicle into a set of equations, such models enable both theoretical analysis and practical implementation of advanced control methods such as model predictive control (MPC), see subsection 2.6.2. Vehicle model complexity can vary considerably; for instance, testing vehicle control systems at the piston level requires highly complex simulations. However, simplified models are often sufficient for control design and scenario-based testing. Vehicle models can be categorised into several levels of complexity. At the most basic level, *kinematic models* describe vehicle motion based purely on geometric constraints, ignoring dynamic effects such as tire forces and inertia. These models are simple, computationally efficient, and widely used for trajectory planning and control at low speeds. In contrast, *dynamic models* incorporate the forces and moments acting on the vehicle, including tire-road interaction, inertia, and load transfer. Dynamic models can be further divided into longitudinal, lateral, and full-vehicle (three- or six-degree-of-freedom) models, depending on the required accuracy. Among these, the *bicycle model* is particularly popular due to its balance between simplicity and realism: it reduces the four-wheel vehicle into an equivalent two-wheel representation, capturing essential lateral dynamics while remaining tractable for control design. High-fidelity models, often based on physics engines or multibody dynamics, include detailed representations of suspension systems, tire models (such as Pacejka's "magic formula"), and

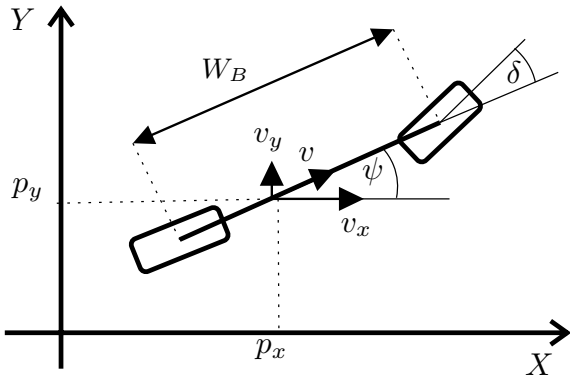


Figure 2.2: Bicycle vehicle model

even aerodynamic effects. While these are indispensable for realistic simulation and vehicle development, they are typically too complex for real-time control algorithms. Therefore, control-oriented modelling seeks a compromise, retaining the dominant dynamics relevant for prediction and decision-making while avoiding unnecessary complexity. In the scope of this thesis, the emphasis is placed on the bicycle model for control-oriented modelling suitable for MPC, and on a dynamic model called the *Multi-Body* vehicle model from the CommonRoad library, developed by Althoff et al. [20], for simulating vehicle movement.

When observed from a top view of a four-wheel vehicle, the bicycle model is obtained by merging the left and right wheels at both the front and the rear into single equivalent wheels. The term bicycle model arises from this abstraction, as the resulting representation resembles a two-wheeled vehicle with a front-wheel steering mechanism analogous to that of a bicycle. Its name implies a two-wheeled configuration, where the front wheel serves solely as a steering component, while the rear wheel assumes a passive role.

Propulsion and braking forces are modelled to act at the vehicle's centre of mass. The global position (p_x, p_y) of the vehicle in a bicycle model can be defined by the following equation:

$$\begin{aligned} \frac{dp_x}{dt} &= v\cos(\psi), \\ \frac{dp_y}{dt} &= v\sin(\psi). \end{aligned} \tag{2.10}$$

Velocity can be described as a function of acceleration through time.

$$\frac{dv}{dt} = a. \tag{2.11}$$

The effects of steering and yaw angle are modelled using the following equation:

$$\begin{aligned} \frac{d\psi}{dt} &= \frac{v}{W_B}\tan(\delta), \\ \frac{d\delta}{dt} &= \dot{\delta}. \end{aligned} \tag{2.12}$$

Now, it is possible to express everything as a state-space system.

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ \mathbf{y} &= \mathbf{h}(\mathbf{x}, \mathbf{u}). \end{aligned} \tag{2.13}$$

where $\mathbf{x}$ is a state vector and $\mathbf{u}$ is the input vector.

$$\mathbf{x} = \begin{bmatrix} p_x \\ p_y \\ v \\ \psi \\ \delta \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} \dot{\delta} \\ a \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{2.14}$$

The complete state-space description of the bicycle model is given as follows:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \end{bmatrix} = \begin{bmatrix} x_3 \cos(x_4) \\ x_3 \sin(x_4) \\ u_2 \\ \frac{x_3}{W_B} \tan(x_5) \\ u_1 \end{bmatrix}. \tag{2.15}$$

To use the bicycle model from Subsection 2.6.1 as the prediction model for MPC, it is necessary to convert it into a discrete version. The discretisation process for a continuous-time system described by the state-space equations in 2.13 can be achieved using the Euler method. The Euler method is a straightforward numerical integration technique used for approximating the solution of ordinary differential equations *ODE* or, in this context, for discretising continuous-time systems. The basic idea is to approximate the derivative by a finite difference and use this approximation to update the state variables over small time intervals. The discrete-time state update can be achieved as given in 2.16:

$$\mathbf{x}^+ = \mathbf{x} + \Delta t \cdot \mathbf{f}(\mathbf{x}, \mathbf{u}). \tag{2.16}$$

In this context, $\mathbf{x}^+$ refers to the state at time step $k+1$, $\mathbf{x}$ represents the state at time step k, Δt is the duration of each time step, $\mathbf{f}$ describes the system dynamicsand $\mathbf{u}_k$ is the input at time step k. By substituting 2.15 into 2.16, the discrete nonlinear bicycle model can be expressed as follows:

$$\mathbf{x}^+ = \begin{bmatrix} x_1 + \Delta t \cdot x_3 \cos(x_4) \\ x_2 + \Delta t \cdot x_3 \sin(x_4) \\ x_3 + \Delta t \cdot u_2 \\ x_4 + \Delta t \cdot \frac{x_3}{W_B} \tan(x_5) \\ x_5 + \Delta t \cdot u_1 \end{bmatrix}. \tag{2.17}$$

Linearisation is a method that approximates nonlinear functions with linear counterparts around a certain operating point. As the distance from the operating point increases, the approximation error also rises. To mitigate the impact of this issue, the common approach is to update the operating point more frequently. Linearisation is commonly performed by utilising the first-order Taylor series expansion around an operating point for a given nonlinear function. The first-order Taylor expansion is given in 2.18.

$$f(x) \approx f(\bar{x}) + f'(\bar{x}) \cdot (x - \bar{x}) + \dots \tag{2.18}$$

Here, $f(\bar{x})$ corresponds to the value of the function at the point $\bar{x}$, while $f'(\bar{x})$ represents the first derivative of the function evaluated at $\bar{x}$. For a multivariate function, such as the

bicycle model given in 2.17, denoted as $\mathbf{f}(\mathbf{x})$, where $\mathbf{x}$ is a vector, the Taylor series expansion around a point $\bar{\mathbf{x}}$ is given by 2.19.

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\bar{\mathbf{x}}) + \nabla \mathbf{f}(\bar{\mathbf{x}}) \cdot (\mathbf{x} - \bar{\mathbf{x}}) + \dots \tag{2.19}$$

In this context, $f(\bar{\mathbf{x}})$ signifies the value of the function at the point $\bar{\mathbf{x}}$, while $\nabla f(\bar{\mathbf{x}})$ denotes the gradient of f evaluated at operating point defined as follows:

$$\overline{\mathbf{x}} = \begin{bmatrix} \overline{x_1} & \overline{x_2} & \overline{x_3} & \overline{x_4} & \overline{x_5} \end{bmatrix}^T. \tag{2.20}$$

substituting $\mathbf{x}^+$ from 2.17 into 2.19 as $\mathbf{f}(\mathbf{x})$, gives linearised system:

$$\mathbf{x}^+ \approx \begin{bmatrix} x_1 + \Delta t \cdot \cos(\overline{x_4}) x_3 - \Delta t \cdot \overline{x_3} \sin(\overline{x_4}) x_4 \\ x_2 + \Delta t \cdot \sin(\overline{x_4}) x_3 + \Delta t \cdot \overline{x_3} \cos(\overline{x_4}) x_4 \\ x_3 + \Delta t \cdot u_2 \\ x_4 + \Delta t \cdot \frac{1}{W_B} (\tan(\overline{x_5}) x_3 + \overline{x_3} \sec^2(\overline{x_5}) x_5) \\ x_5 + \Delta t \cdot u_1 \end{bmatrix}. \tag{2.21}$$

Here, elements $x_1, x_2, \dots, x_5$ represent small offsets from the operating point. The matrix $\mathbf{A_L}$ for the linearised version of the vehicle bicycle model can be written as follows 2.22:

$$\mathbf{A_L} = \begin{bmatrix} 1 & 0 & \Delta t \cos(\overline{x_4}) & -\Delta t \cdot \overline{x_3} \sin(\overline{x_4}) & 0 \\ 0 & 1 & \Delta t \sin(\overline{x_4}) & \Delta t \cdot \overline{x_3} \cos(\overline{x_4}) & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \Delta t \frac{\tan(\overline{x_5})}{W_B} & 1 & \Delta t \frac{\overline{x_3} \sec^2(\overline{x_5})}{W_B} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.22}$$

The matrix $\mathbf{B_L}$ is given as follows:

$$\mathbf{B_L} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \Delta t \\ 0 & 0 \\ \Delta t & 0 \end{bmatrix}. \tag{2.23}$$

The system can now be written as

$$\dot{\mathbf{x}} = \mathbf{A_L}\mathbf{x} + \mathbf{B_L}\mathbf{u}. \tag{2.24}$$

Although simplified, the bicycle model shown in Fig. 2.2 provides reliable future state estimations while maintaining high computational efficiency. In this thesis, both the linearised and nonlinear vehicle models are implemented in Python [50], relying on the NumPy package [51] for numerical computations. The nonlinear model has been implemented in Python.

2.6.2 *Model Predictive Control*

MPC is a class of advanced control strategies that has gained widespread use in both academic research and industrial applications since its introduction in the late 1970s [52, 53]. The main concept behind MPC is the repeated solution of an optimal control problem over a

finite prediction horizon. At each discrete time step, the controller employs a mathematical model of the system to predict future trajectories of the states and optimises the control inputs by minimising a cost function that accounts for tracking performance and control effort. This yields control values for the next number of steps equal to the size of the prediction horizon. Only the first input of the optimised sequence is applied to the plant, and the optimisation is repeated at the next step in a receding horizon fashion [53, 54]. This predictive and iterative nature distinguishes MPC from classical controllers such as Proportional–Integral–Derivative (PID) controller or linear state-feedback approaches.

A fundamental advantage of MPC lies in its ability to explicitly incorporate system constraints into the optimisation problem. Constraints can include actuator limits, safety requirements, or operational boundaries that must not be violated during operation. For automotive systems, this capability is particularly important, as real vehicles must always remain within physical limitations of braking, steering, and tire-road interactions [55, 20]. Whereas conventional controllers often require ad-hoc measures to avoid constraint violations, MPC naturally integrates them into the optimisation, providing a principled approach to constraint handling. In practice, the quality of MPC depends strongly on the fidelity of the underlying prediction model. Automotive applications often employ the bicycle model [17], described in Subsection 2.6.1. Depending on the application, either a linearised or a nonlinear bicycle model can be used. Accordingly, MPC can be classified into Linear MPC (LMPC) and Nonlinear MPC (NMPC). NMPC employs the full nonlinear model of the vehicle, providing higher accuracy and robustness during aggressive manoeuvres. However, this accuracy comes at considerable computational cost, which may limit real-time feasibility in embedded systems. LMPC, on the other hand, relies on local linearizations around the current operating point, reducing computational demands at the expense of accuracy [56, 57]. This trade-off between fidelity and efficiency is central to MPC research, especially in safety-critical applications where both reliability and fast computation are required. The optimisation problem solved by MPC is typically formulated as a quadratic cost function that penalizes deviations from the reference trajectory and large variations in control inputs. A standard formulation is given as [52, 55]:

$$J = \sum_{k=0}^{N-1} \left[e_k^T Q e_k + \Delta u_k^T R \Delta u_k \right] + e_N^T S_N e_N, \tag{2.25}$$

where e_k denotes the state tracking error at step k, Q and R are weighting matrices for state and control input penalties, Δu_k is the change in control input, and S_N is the terminal weighting matrix applied to the error at the end of the horizon. This quadratic structure ensures that, for linear models with convex constraints, the optimisation problem is a convex quadratic program, which can be solved efficiently using existing solvers.

In the automotive domain, MPC has been applied to a wide range of tasks, including trajectory tracking, path following, collision avoidance, adaptive cruise control, and stability enhancement [55, 20, 17]. In the scope of this thesis, MPC is employed for the novel filtering method described in Chapter 3.

2.7 ARTIFICIAL DISTURBANCES

This section describes how commonly encountered disturbance types were mimicked and introduced in the dataset. From the GT, the x and y positions, along with a time vector, were selected for adding the distortions. Three types of distortions were introduced to the data: noise, random sample removal, and random outlier injection.

2.7.1 *Noise*

Noise was generated by applying a specific noise-to-signal ratio (NSR) to the ground truth (GT) data, ensuring controlled degradation of signal quality. Additionally, the noise could vary in type, either white or coloured.The process of adding noise to the signal begins with calculating the signal power. The average power of the signal, P_{signal}, is computed using the formula:

$$P_{\text{signal}} = \frac{1}{n}\sum_{i=1}^{n}(x_i)^2, \tag{2.26}$$

where x_i represents the samples of the signal, and n is the total number of samples. Using the noise-to-signal ratio NSR, the noise power P_{noise} is calculated as:

$$P_{\text{noise}} = P_{\text{signal}} \times 10^{\left(\frac{\text{NSR}_{\text{dB}}}{10}\right)}. \tag{2.27}$$

White noise is then generated as a normal distribution with zero mean and standard deviation $\sqrt{P_{\text{noise}}}$.Finally, the generated noise was added to the original signal to create the noisy version.

2.7.2 *Outliers*

Outliers were introduced to simulate extreme deviations in the data, which are commonly encountered in real-world signals due to sensor errors or unexpected events. The outliers added to the GT data were generated by adding values several times larger than the average signal value at randomly selected samples. The number of outliers, N_o, was determined by multiplying a specified percentage O_p with the total number of samples N_s and rounding the result to the nearest whole number. The amplification of the added values is controlled by a factor O_A, which scales the average signal value. Each outlier was introduced by adding a multiple of the average value of the corresponding data entry. The sign of the outliers was randomized to simulate both positive and negative deviations. Additionally, the values of the outliers varied to prevent uniform patterns, ensuring the deviations mimicked realistic scenarios.

2.7.3 *Irregular Sampling*

Irregular sampling was simulated to mimic data gaps or uneven sampling rates, which commonly occur due to communication failures, low sensor resolution, or incomplete data acquisition. This was achieved by removing a specified percentage r_p of samples from each signal. The number of samples to be removed, N_r, was calculated by multiplying the specified percentage r_p by the total sample count N_s and rounding to the nearest whole

number. Random indices were then selected to identify which samples to remove, ensuring that no sample was selected more than once. This method introduced gaps in the data, effectively replicating missing samples or irregularly spaced data points observed in real-world scenarios.

2.8 RAUCH TUNG STRIEBEL SMOOTHER

RTS smoother is a classical offline state estimation method that extends the Kalman filter by conditioning state estimates on the complete measurement sequence, including both past and future observations [58]. In contrast to the recursive Kalman filter, which provides causal state estimates based solely on past measurements, the RTS smoother performs an additional backward pass that refines these estimates and produces smoother, more temporally consistent trajectories.

The RTS algorithm operates in two stages. In the first stage, a forward Kalman filter is applied to the discrete-time system model

$$x_{k+1} = A_k x_k + B_k u_k + w_k, \tag{2.28}$$

with the measurement model

$$y_k = C_k x_k + v_k, \tag{2.29}$$

where x_k denotes the system state, u_k the control input, w_k the process noise, and v_k the measurement noise. The forward filtering step yields the filtered state estimate $\hat{x}_{k|k}$ and its covariance $P_{k|k}$ according to

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \left(y_k - C_k \hat{x}_{k|k-1} \right), \tag{2.30}$$

where K_k is the Kalman gain.

In the second stage, the backward RTS recursion refines the forward estimates using future information. Starting from the final time step N, the smoothed state estimate $\hat{x}_{k|N}$ is computed as

$$\hat{x}_{k|N} = \hat{x}_{k|k} + G_k \left(\hat{x}_{k+1|N} - \hat{x}_{k+1|k} \right), \tag{2.31}$$

where the smoothing gain G_k is defined as

$$G_k = P_{k|k} A_k^\top P_{k+1|k}^{-1}. \tag{2.32}$$

This formulation ensures that each state estimate is optimally conditioned on the full measurement set $\{y_1, \ldots, y_N\}$, under the linear-Gaussian assumption.

Since the RTS smoother relies on future measurements, it is inherently non-causal and therefore applicable only in offline processing. In vehicle trajectory reconstruction, this makes it suitable for post-processing recorded driving data, where it effectively suppresses measurement noise and enforces temporal smoothness. When combined with a physically meaningful vehicle model consistent with the MPC formulation used in this thesis, the RTS smoother produces continuous state trajectories without abrupt discontinuities.

However, the RTS smoother optimizes statistical consistency rather than control or feasibility objectives over an extended horizon. Unlike MPC, it does not explicitly incorporate future objectives, control constraints, or optimisation over a finite prediction horizon.

As a result, dynamically relevant short-duration manoeuvres may be attenuated, and the reconstructed trajectory may not be optimal for downstream simulation or control-based evaluation.

In this thesis, the RTS smoother is used as a reference method for trajectory filtering. Its performance is evaluated under the same system model, state representation, and disturbance assumptions as the proposed extended-horizon MPC-based filtering approach, enabling a direct and fair comparison.

2.9 SUMMARY

This chapter established the conceptual and technical foundations used throughout the thesis. It introduced scenario-based testing and the standard scenario abstraction levels (functional, logical, and concrete), with emphasis on data-driven logic scenarios and the need for automated conversion from recorded trajectories. It then presented the curvilinear coordinate system (Section 2.3) as a road-aligned representation that supports manoeuvre description and classification, and reviewed polynomial and spline-based approximation methods, including Greedy Knot Reduction (Subsection 2.4.2) as a mechanism for controlling approximation complexity. The chapter also summarised relevant machine learning concepts and neural network architectures (Section 2.5) used for manoeuvre recognition. Finally, it formulated control-oriented vehicle models based on the bicycle abstraction and derived their discrete and linearised forms (Subsection 2.6.1), introduced the MPC formulation (Subsection 2.6.2), and defined the disturbance models and RTS smoother baseline (Sections 2.7 and 2.8) used later for consistent evaluation.

Together, these elements provide the groundwork for longitudinal and lateral action classification (Sections 5.3 and 5.2) and for the MPC-based filtering framework in Chapter 3.

3

Extended Horizon Model-predictive Trajectory Estimation and Parameter Estimation

This chapter presents a novel vehicle trajectory filtering method based on model simulation, earlier introduced in the author's work [59]. Rather than applying conventional filtering, the method reconstructs the trajectory by simulating a vehicle model that tracks the input trajectory as a reference path. The output of the filtering method is an approximated trajectory with a higher sampling resolution, limited only by the simulation sampling rate. It is free of disturbances and includes complete vehicle states and the control inputs of the simulated model.

3.1 Introduction

The filtering method introduced in this section is based on an extended horizon model predictive controller that tracks the input trajectory as the path for a simulated vehicle model to follow. Standard MPC, described in Subsection 2.6.2, is modified by extending its horizon beyond the target state. The applied modifications are presented in Subsection 3.2.1. This modification allows it to provide an optimal control sequence to achieve the current desired state and anticipate the next goal state. The inspiration for this modification originates from the task of navigating a vehicle through a cone obstacle course. While the driver focuses primarily on steering around the immediate cone, they are simultaneously aware of and preparing for the next cone. Thus, attention is centred on the current obstacle while still anticipating those that follow. The method is divided into two distinct steps: preprocessing and filtering simulation, as shown in Fig. 3.1. The preprocessing step converts the input trajectory into estimated vehicle model states required by the MPC. Since the input trajectory consists only of vehicle positions (X, Y) over time T, the states are estimated using basic kinematic formulas described in Section 3.2. Once the input trajectory is expanded with estimated vehicle states, the second step of filtering can begin. In this stage, the modified MPC filter uses the reconstructed states as a sequence of goal states that the simulated vehicle must follow. Two vehicle models are required: a simplified vehicle model for MPC-based state estimation, and a high-fidelity vehicle model that approximates the dynamics of the real vehicle that generated the input trajectory. The process applies control inputs obtained from the modified MPC to drive the complex vehicle model in simulation so that it follows the reconstructed states. After the filtering simulation is completed, the results consist of vehicle states and control inputs. A filtered vehicle trajectory can be extracted

from the vehicle states.

The capability of the extended-horizon MPC to estimate vehicle parameters is covered in the last section of this chapter (Section 3.3). The evaluation is performed by estimating vehicle mass on both the original HighD trajectories and their simulated counterparts obtained from the converted logic scenarios. These evaluations are presented in Subsections 3.3.1 and 3.3.2, respectively.

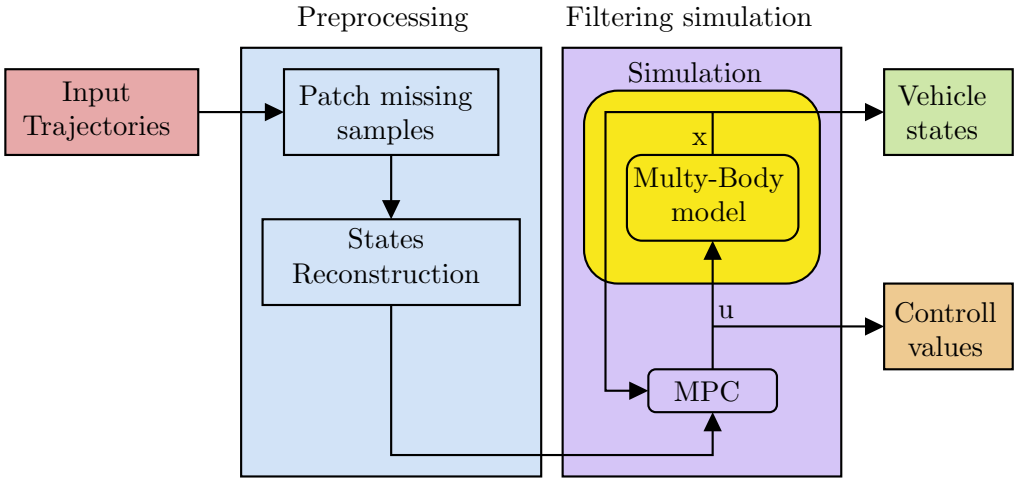


Figure 3.1: Overview of MPC-based filtering steps

3.2 PATCHING AND RECONSTRUCTION OF VEHICLE STATES

The input trajectory consists of three vectors: X and Y, representing vehicle position, and T, representing time. Given the presence of noise, disturbances, missing samples, or uneven sampling in the trajectory, proper preprocessing is required. The most critical issue is missing or uneven samples. The first step involves detecting missing samples and interpolating them to maintain uniform sampling. This is achieved by analysing the time intervals between consecutive sample points.A reference sample time interval, denoted as Δt_{ref}, is determined by identifying the smallest consistent time difference across the input data:

$$\Delta t_{\text{ref}} = \min(T_{i+1} - T_i). \tag{3.1}$$

If the gap between two samples exceeds Δt_{ref}, missing samples are detected. The number of missing samples is estimated as:

$$\text{Number of missing samples} = \left\lfloor \frac{\Delta t}{\Delta t_{\text{ref}}} \right\rfloor - 1, \tag{3.2}$$

where $\Delta t = T_{i+1} - T_i$. The next step involves reconstructing the data at missing points using linear interpolation. For a missing sample at index k, the interpolated values for the X and Y coordinates are calculated as:

$$X_k = X_i + \frac{k-i}{n}(X_{i+1} - X_i), \tag{3.3}$$

$$Y_k = Y_i + \frac{k-i}{n}(Y_{i+1} - Y_i), \tag{3.4}$$

where n is the total number of missing samples plus one, and k runs from 1 to n. The time vector is reconstructed as:

$$T_k = T_i + k \cdot \Delta t_{\text{ref}}. \tag{3.5}$$

This linear interpolation technique generates new values for missing samples, ensuring a smooth and continuous transition across the gap.

After patching, the next stage reconstructs the vehicle's states using basic vehicle kinematics. This process begins with calculating the vehicle velocity from the patched spatial and temporal data. Velocity is calculated as the rate of change of the vehicle position over time, which requires accurate and consistent time intervals between data points. This is expressed as:

$$v(t) = \frac{\sqrt{(x(t) - x(t-\Delta t))^2 + (y(t) - y(t-\Delta t))^2}}{\Delta t}. \tag{3.6}$$

Here $x(t)$ and $y(t)$ denote the spatial coordinates at time t, and Δt represents the time interval between consecutive samples. The vehicle's heading angle is computed from the directional changes observed in its trajectory. This angle indicates the vehicle's orientation relative to its previous position:

$$\theta(t) = \text{atan2}(y(t) - y(t-\Delta t), x(t) - x(t-\Delta t)), \tag{3.7}$$

The steering angle of the vehicle is calculated as:

$$\delta(t) = \text{atan2}\left(\frac{L \cdot \left(\frac{\theta(t) - \theta(t-\Delta t)}{\Delta t}\right)}{v(t)}\right). \tag{3.8}$$

Here $\delta(t)$ denotes the steering angle at time t, L is the wheelbase (the distance between the front and rear axles), and $\dot{\theta}(t)$ represents the yaw rate (rate of change of the heading angle). Additionally, the change in heading angle, often referred to as the angular rate or yaw rate, is calculated to provide further insights into the vehicle's steering and turning behaviour. To ensure smoothness and accuracy in estimating these vehicle state variables, a low-pass filter can be pre-applied to the data. This filter reduces high-frequency noise that could distort velocity and heading angle calculations, enabling a more reliable reconstruction of the vehicle's dynamic states.

The result of this section is a comprehensive vehicle state profile, including the vehicle's position, velocity, heading angle, and changes in heading angle over time. These vehicle states are used for the next step of simulation-based filtering, described in the next section.

3.2.1 *MPC Modification for Filtering*

For filtering applications with the modified MPC, the input trajectory becomes a sequence of goal states that the vehicle must track, also referred to as future states. The standard MPC cost function, given in Equation 2.25, cannot accommodate such a list of future states. To address this, an additional term is introduced into the cost function:

$$J_{Ext} = \sum_{j=0}^{M} [(x_{N+j} - X'_j)^T \frac{\mathbf{FN}}{f_z(j)} (x_{N+j} - X'_j)]. \tag{3.9}$$

where **FN** is a weighting matrix for future states, while $f_z(j)$ is a decay function that gradually reduces the weight of future states the further into future they are. A typical choice for $f_z(j)$ is an exponential decay, $f_z(j) = z^j$, although any monotonically decreasing function can be applied. The term $x_{N+j} - X'_j$, analogous to e_N, represents the error between the predicted states and the reference future states. While the future look horizon, denoted by M, specifies the number of future states included in the expression. Now the cost function with added J_{Ext} member and J_0 from Equation 2.25 can be expressed as:

$$J_N = J_0 + J_{Ext}. \tag{3.10}$$

Finally, the complete version of the modified cost function can be expressed as:

$$J_N = \sum_{k=0}^{N-1} \left[e_k^T Q e_k + \Delta u_k^T R \Delta u_k \right] + e_N^T S_N e_N + \sum_{j=0}^{M} [(x_{N+j} - X'_j)^T \frac{\mathbf{FN}}{f_z(j)} (x_{N+j} - X'_j)]. \tag{3.11}$$

A more graphical description of this approach can be seen in Fig. 3.2. In addition to the modified cost function, the prediction horizon N needs to be decreased by one at each step towards the current goal state X_1 and reset to N once the goal is reached. Solving the cost function provides the optimal control sequence for reaching the goal state within N steps. However, since each step moves the vehicle model in simulation closer to the current goal, the number of remaining steps within the fixed time frame decreases. This adaptive adjustment of the prediction horizon ensures consistency between sampling time and remaining steps. Similar modifications of MPC have been proposed and evaluated

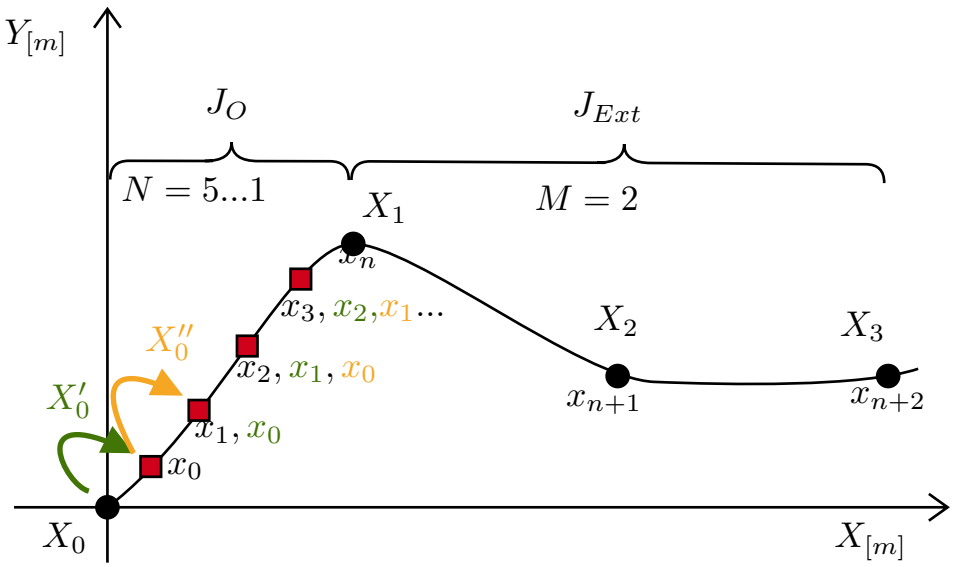


Figure 3.2: Modified MPC filter concept

in [60] and [61] for UAV navigation and obstacle avoidance. In those works, the MPC formulation is used as a real-time trajectory tracking component that produces smooth and dynamically feasible motion while reacting to online path updates caused by obstacle avoidance. In contrast, the approach presented in this thesis uses MPC as an offline filtering and reconstruction tool, where the measured trajectory is treated as a sequence of goal states and the simulated vehicle is driven through the data to obtain a physically consistent,

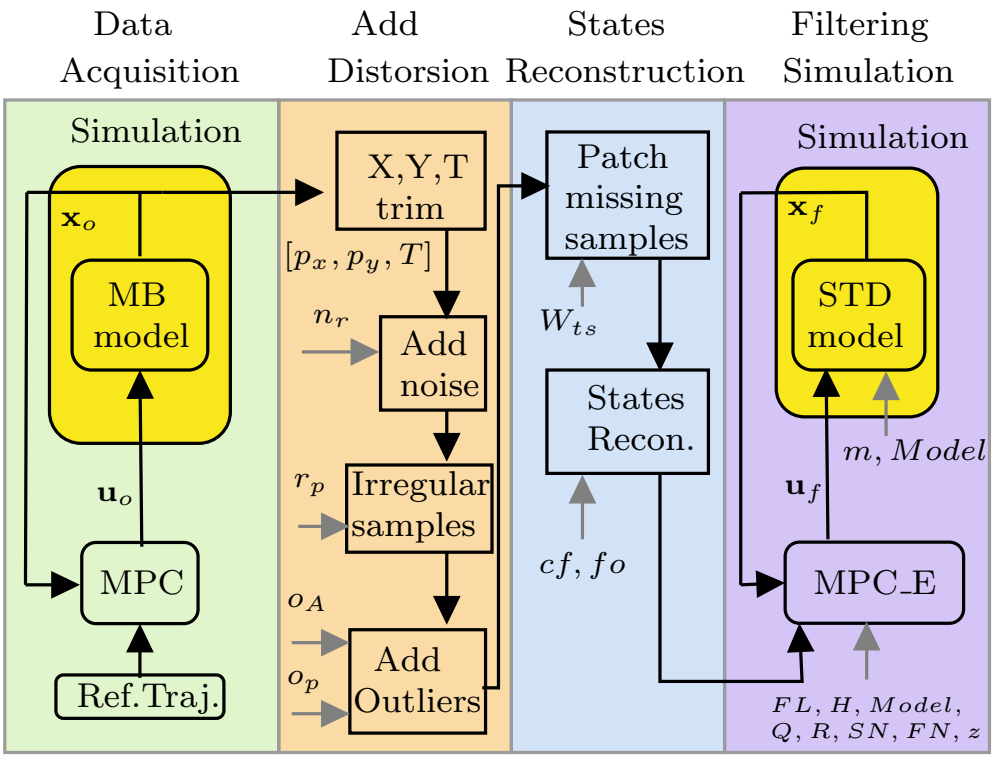


Figure 3.3: Evaluation configuration overview diagram

disturbance-reduced trajectory together with the corresponding vehicle states and control inputs.

For accurate filtering, it is necessary to have a vehicle model as close as possible to the original vehicle that produced the input trajectory. For more precise filtering results, the simulation can incorporate additional factors such as weather conditions, road type, surface properties, terrain inclination, etc.

3.2.2 *Evaluation*

The evaluation was conducted in two steps. In the first step, the goal was to identify the optimal configuration for the given parameters: FL, Z, and horizon size. The second step examined how variations in vehicle mass in the simulation influenced the results. Additionally, the same trajectories were filtered using the RTS smoother, introduced in Section 2.8, for performance comparison.

Filtering simulations were performed using the Single Track Drift (STD) model [20], a modified single-track model with an introduced mass. Tire forces were calculated using the Pacejka tire model [62]. The STD model was used for filtering because it offers the best compromise between computational efficiency and simulation accuracy compared to more complex vehicle models. A general overview of the evaluation configuration is shown in the diagram in Fig. 3.3. The configuration and simulation were implemented in Python. The *CasADi* symbolic framework for Python [63] was used to define the optimisation problems, while the Interior Point Optimizer *Ipopt* [64] was employed to solve them. An identical vehicle simulation configuration was used in earlier work [65], where the focus was on path

following and MPC implementation. Here, the goal is to extend the MPC formulation to act as a filter by applying the computed control inputs to a more complex vehicle model. This effectively makes the vehicle "drive" through the measurement points and filter the data. Moreover, in that work the reference path was clearly defined, whereas here the trajectories are reconstructed from noisy data with missing samples.

With the stated goal of identifying the optimal parameter configuration, multiple experimental runs were conducted using both SLC and DLC trajectories [55, 20], each subjected to varying levels of applied distortion. The DLC follows the standardized double lane-change manoeuvre definition commonly used for vehicle dynamics and control benchmarking, while the SLC represents the corresponding single lane-change variant generated using the same trajectory-shaping parameterisation. Both trajectories were generated in simulation at a constant speed of 5 m/s, using the parameter values listed in Table 3.1 and the following expressions:

$$y_{\text{ref}} = \frac{d_{y_1}}{2}\left(1 + \tanh(z_1)\right) - \frac{d_{y_2}}{2}\left(1 + \tanh(z_2)\right) \tag{3.12}$$

$$\psi_{\text{ref}} = \arctan\left(\frac{d_{y_1} \cdot 1.2}{d_{x_1} \cdot \cosh(z_1)^2} - \frac{d_{y_2} \cdot 1.2}{d_{x_2} \cdot \cosh(z_2)^2}\right) \tag{3.13}$$

$$z_1 = \frac{shape}{d_{x_1}}\left(X - X_{s_1}\right) - \frac{shape}{2} \tag{3.14}$$

$$z_2 = \frac{shape}{d_{x_2}}\left(X - X_{s_2}\right) - \frac{shape}{2} \tag{3.15}$$

These expressions were used to obtain the reference trajectories that served as inputs for vehicle motion simulation. The resulting trajectories are shown in Fig. 3.4.

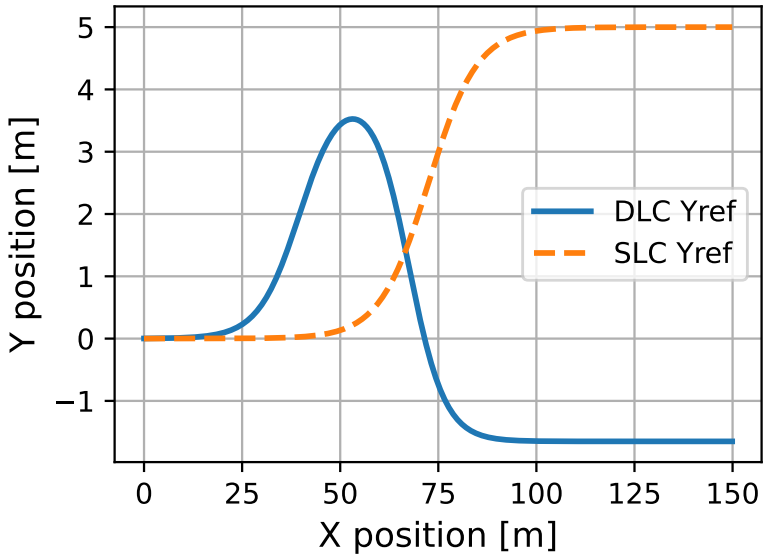


Figure 3.4: a plot of DLC and SLC trajectories

The values of parameter FL were varied from 0 to 25, the Z parameter was tested at values of 0.9, 0.97, 1, 1.05, 1.1, 1.2, 1.5, and 2, while the prediction horizon was evaluated for three sizes: 6, 10, and 15. As a control, a distortion-free trajectory was also filtered to

Table 3.1: DLC and SLC parameters

Parameter	*shape*	d_{x_1}	d_{x_2}	d_{y_1}	d_{y_2}	X_{s_1}	X_{s_2}
DLC	2.4	25	21.95	4.05	5.7	27.19	56.46
SLC	2	25	21.95	5	5.7	60	1000

Table 3.2: Configuration parameters and results: SLC

ID		**1**	**2**	**3**	**4**	**5**
Parameter	n_r	0.1	0.1	0.2	0.5	0
	r_p	5%	5%	20%	20%	0%
	O_A	2%	10%	10%	20%	0
	O_p	2	5	10	10	0
Best	FL, z, h	8, 0.9, 15	11, 1.2, 15	2, 0.9, 15	4, 0.95, 6	18, 1.1, 6
	e_{pos}	**0.00503**	**0.03340**	**0.17960**	**0.28142**	0.00046
	σ_{pos}	**0.00106**	**0.00621**	**0.00629**	**0.01487**	**0.00001**
RTS	e_{pos}	0.00677	0.13212	1.30350	1.42003	**0.00003**
	σ_{pos}	0.01077	0.32903	2.30069	2.09224	0.00006

determine the extent of distortion introduced by the filtering method itself. Vehicle mass was set to be the same as MB model in reference simulation $m = 1000\ kg$. Distortion parameters and results are presented in Tables 3.2 and 3.3. This table contains results for the three best parameter sets for each configuration, along with RTS. The mean square error between the reference vehicle trajectory and the filtering results is denoted by e_{pos}, while the standard deviation of the same values is represented as δ_{pos}. A visual representation of the distorted trajectories used as the filter input can be seen in Figs. 3.5a and 3.5b. Filtering performance in terms of MSE position, in relation to parameters FL, Z, and the horizon, can be seen in Fig. 3.6. Filtering results, together with the RTS results, are shown in Figs. 3.7a and 3.7b. The three best overall configurations for the combined SLC and DLC cases were achieved with $FL = 1$ and a horizon size of 15. The main difference was in the Z values, with the best result at $Z = 1.2$, the second-best at $Z = 1.5$, and the third-best at $Z = 0.95$.

The three configurations that demonstrated the best overall performance in the first part were used in the second part of the evaluation, with variable vehicle masses in the filtering simulation. For each filtering simulation, the vehicle mass was varied with values of 500, 1000,1500, 2000, 3000, 4000, 5000, and 6000 kg. Distortion parameters were set to configuration ID one from Tables 3.2 and 3.3. Filtered trajectories are shown in Figs. 3.7a and 3.7b, and the corresponding values of error and deviation can be seen in Table 3.4. The execution time of the proposed method depends on the FL variable, used simulator, and the horizon size, ranging from approximately 100 s to longer, whereas the RTS smoother processes the aforementioned data in a bit less than 0.3 s. However, note that the proposed method is intended to be used offline in a single run to preprocess the data.

Table 3.3: Configuration parameters and results: DLC

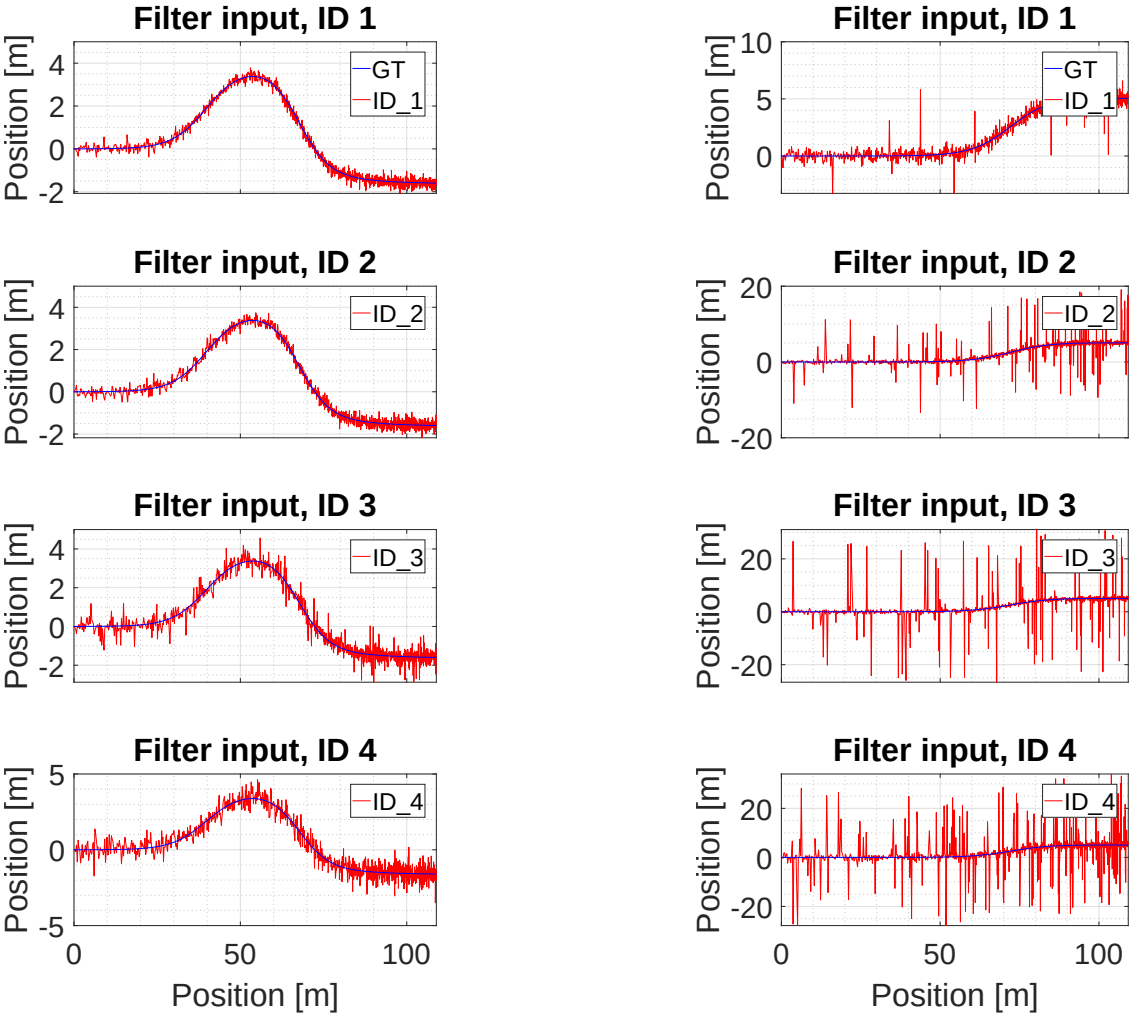
ID		**1**	**2**	**3**	**4**	**5**
Parameter	n_r	0.1	0.1	0.2	0.5	0
	r_p	5%	5%	20%	20%	0%
	O_A	2%	10%	10%	20%	0
	O_p	2	5	10	10	0
Best	FL, z, h	8, 1.1, 10	11, 1.05, 6	5, 1.1, 6	6, 1, 6	13, 1.2, 15
	e_{pos}	0.02546	0.01110	0.02104	0.02438	0.02579
	σ_{pos}	**0.0027**	0.00226	**0.00451**	**0.00551**	0.00155
RTS	e_{pos}	**0.00117**	**0.00194**	**0.00498**	**0.00751**	**0.00018**
	σ_{pos}	0.00155	**0.00212**	0.00684	0.00915	**0.00025**

Table 3.4: Comparison of mass influence on MSE and STD position metrics.

mass	z=0.95		z=1.2		z=1.5	
$[kg]$	e_{pos}	σ_{pos}	e_{pos}	σ_{pos}	e_{pos}	σ_{pos}
500	0.03465	0.06094	**0.03241**	0.05837	**0.03205**	0.05621
1000	**0.034**	**0.05974**	0.03338	**0.05801**	0.03283	**0.05581**
1500	0.03826	0.06757	0.03952	0.06768	0.04059	0.06809
2000	0.03869	0.06983	0.03991	0.07073	0.04153	0.072
3000	0.03853	0.07177	0.03998	0.07387	0.0412	0.07541
4000	0.03737	0.06999	0.03853	0.07176	0.03986	0.073
5000	0.03625	0.06875	0.03707	0.07	0.0379	0.07111
6000	0.03526	0.06806	0.0358	0.06916	0.0365	0.07017

3.3 VEHICLE PARAMETER ESTIMATION

In addition to its filtering capability, the proposed approach can be extended to the estimation of vehicle parameters such as mass, wheelbase, and track width. This is achieved by performing multiple simulations using the same input trajectory while varying the parameter configuration of the simulation model. The underlying assumption is that discrepancies between the simulated and measured trajectories indicate a mismatch between the assumed and actual vehicle parameters. For example, if the simulated vehicle mass is significantly higher than that of the real vehicle from which the trajectory data were obtained, deviations become apparent during directional changes, as the heavier model responds more sluggishly to steering inputs. Conversely, when the simulated mass is substantially lower than the true vehicle mass, discrepancies are most pronounced in driving segments involving rapid acceleration or braking, where the lighter model exhibits stronger dynamic responses than those observed in the reference trajectory. By minimising these deviations across different driving regimes—such as cornering, acceleration, and braking—the parameter set of the simulated vehicle can be adjusted to approximate the physical characteristics of the real vehicle. For evaluation, vehicle mass estimation is performed in two cases, as shown in Fig. 3.8.



(a) DLC

(b) SLC

Figure 3.5: Input Trajectories

In the first case covered in subsection 3.3.1, mass is estimated from a trajectory with unknown vehicle mass obtained from the HighD dataset. In the second case, described in Subsection 3.3.2, mass estimation is performed on a trajectory with known vehicle mass, generated in the dSPACE ASM simulation environment [66].

3.3.1 *Evaluation on HighD Trajectories*

To evaluate the vehicle mass estimation capability of the extended horizon MPC framework introduced earlier in this chapter, ten representative trajectories were selected from the HighD dataset [67]. These trajectories were chosen based on their performance in the logic scenario reconstruction and resimulation experiments described in Section 5.5. The HighD dataset does not provide ground-truth information regarding vehicle mass. Instead, it includes a coarse visual classification of vehicles into two categories: *Car* and *Truck*. This classification is derived solely from visual classification of the recorded video data and therefore provides only limited information regarding the true physical properties of the

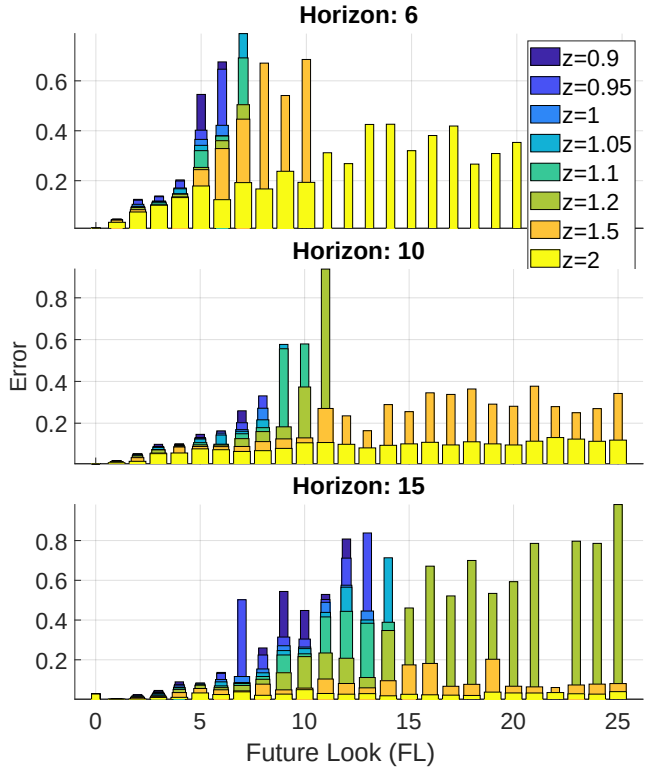
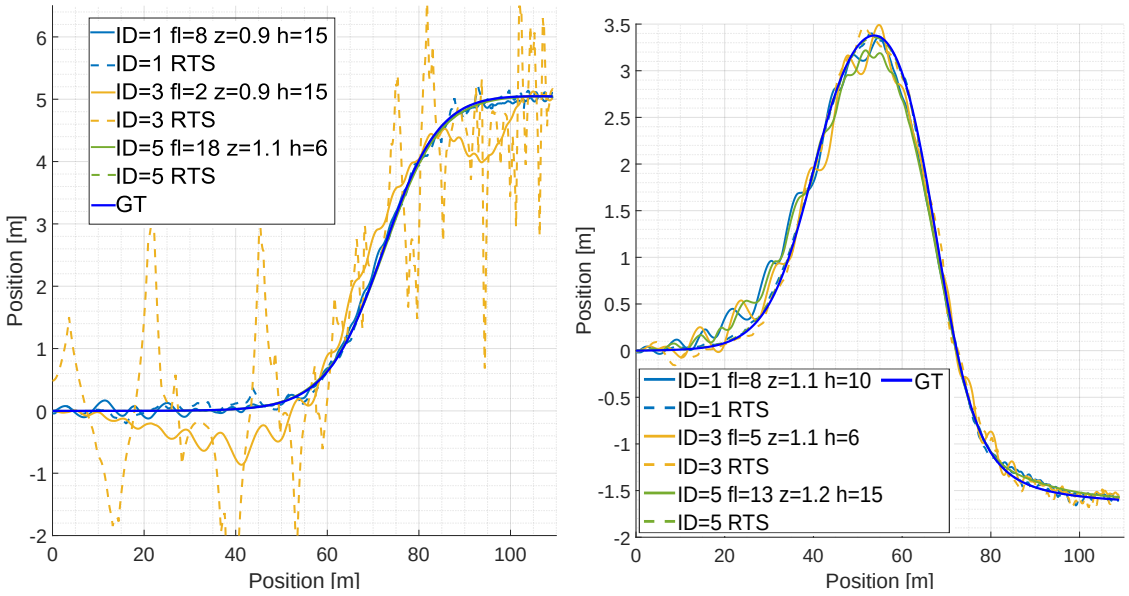


Figure 3.6: Pos. MSE for different FL, Z, and horizon values.

vehicles. The *Car* category encompasses a broad range of vehicle types, from small passenger cars with masses below 1000 kg to larger sport utility vehicles with masses approaching above 2500 kg. Likewise, the *Truck* category exhibits substantial variability due to differences in base vehicle mass as well as variations in carried payload.

To account for this uncertainty, a parametric sweep over candidate vehicle masses was performed in the range from 800 kg to 2700 kg, using increments of 100 kg. The selected HighD trajectories were treated as reference paths that a simulated vehicle was required to follow. For each candidate mass value, an independent filtering and simulation run was executed. For this evaluation, the parameter FL was fixed to 1, while the parameter Z was set to 1.5. The simulation was performed using a sampling time of 1 ms. This small sampling interval was selected to ensure high numerical accuracy and to minimise discretisation effects in the resulting trajectory estimates. The final vehicle mass estimate was determined as the candidate mass that minimised the MSE between the simulated trajectory and the corresponding HighD reference trajectory. Table 3.5 and Figs. 3.9–3.13 provide an overview of the evaluation results. The table reports the standard deviation (STD) and MSE for each simulation, corresponding to the best-matching configuration. The HighD ID denotes the HighD recording segment and the corresponding trajectory identifier of the source vehicle. The Class column presents the HighD vehicle class label for each trajectory.

From the evaluation results, it can be determined which mass estimates are plausible and which are erroneous. Most estimates are consistent with the MSE-versus-mass curves, as the selected mass typically corresponds to a local minimum. The result for ID 8 would likely improve if the evaluated mass range were extended, because the optimum does not lie at a clear local minimum within the tested range. Given that this vehicle is labelled as a car,



(a) SLC results.

(b) DLC results.

Figure 3.7: SLC and DLC results.

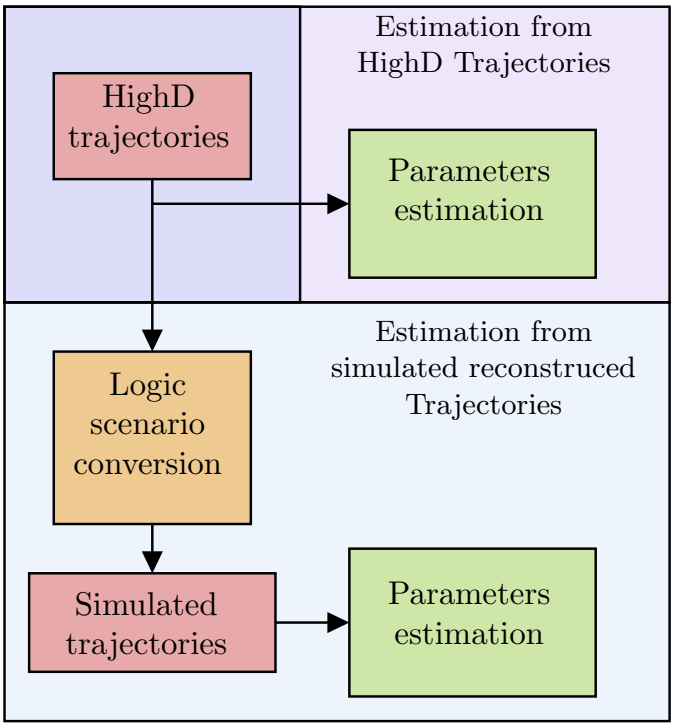
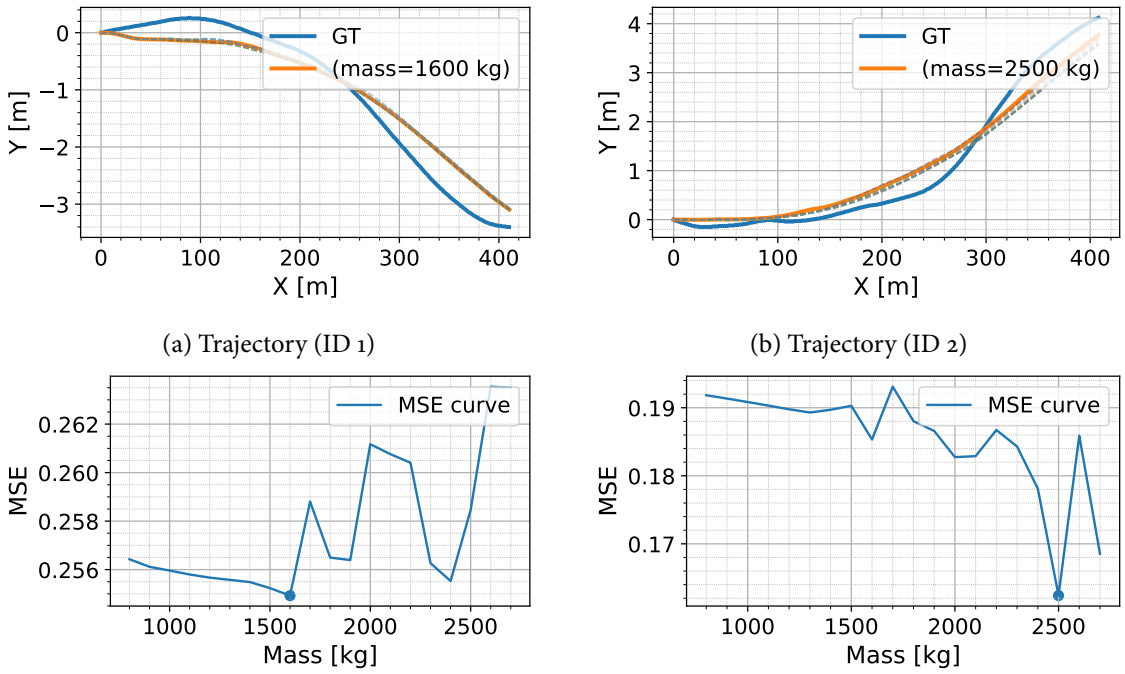


Figure 3.8: Vehicle mass estimation evaluation diagram

the estimate may be implausible, since passenger vehicles typically do not exceed 2700 kg. This behaviour is also consistent with limited manoeuvre excitation, because mass effects are most observable during acceleration, braking, and steering transients.

Table 3.5: HighD vehicle mass estimation results

ID	highD name	std	MSE	est. mass [kg]	class
1	50_1329	0.16482	0.25493	1600	Car
2	51_1106	0.10706	0.16239	2500	Truck
3	51_1139	0.23910	0.62290	2200	Car
4	52_1542	0.15618	0.22099	1300	Car
5	52_696	0.24865	0.13636	2500	Truck
6	54_487	0.15480	0.41009	2100	Truck
7	54_77	0.14701	0.32946	2400	Car
8	55_2086	0.29944	0.43460	2700	Car
9	56_507	0.11460	0.28510	2500	Truck
10	60_1332	0.21045	0.34526	1400	Truck



(a) Trajectory (ID 1)

(b) Trajectory (ID 2)

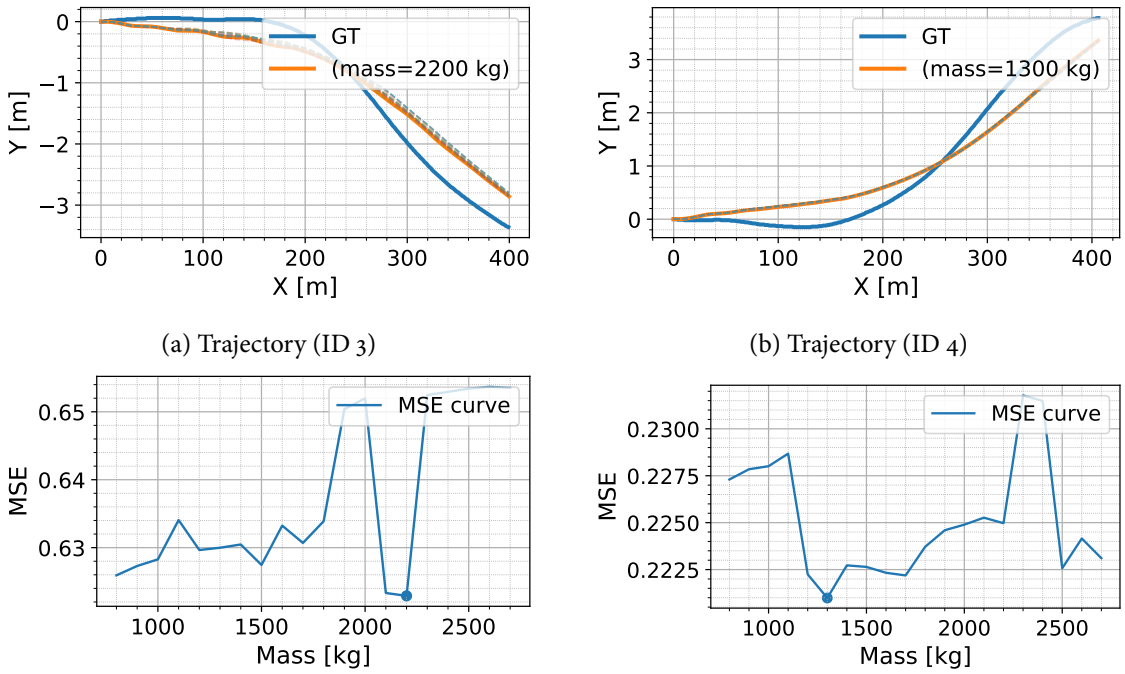
(c) MSE (ID 1)

(d) MSE (ID 2)

Figure 3.9: Mass estimation results for IDs 1 and 2.

3.3.2 *Evaluation on Logic Scenario Simulation*

In contrast to the estimation presented in Subsection 3.3.1, this subsection considers trajectories for which the vehicle mass is known. For all simulations in this subsection, the vehicle mass was fixed at 1500 kg. The trajectories used for this evaluation were generated by simulating logic scenarios derived from the HighD dataset, with the mass of the simulated vehicle explicitly set to 1500 kg. All other parameters of the mass estimation procedure were kept identical to those used in the previous evaluation. The vehicle velocity profile follows the original trajectory and typically lies in the range of 90–120 km/h. Specifically, parameter FL was set to 1, while parameter Z was set to 1.5. The simulation sampling time was set to 1 ms.



(a) Trajectory (ID 3)

(b) Trajectory (ID 4)

(c) MSE (ID 3)

(d) MSE (ID 4)

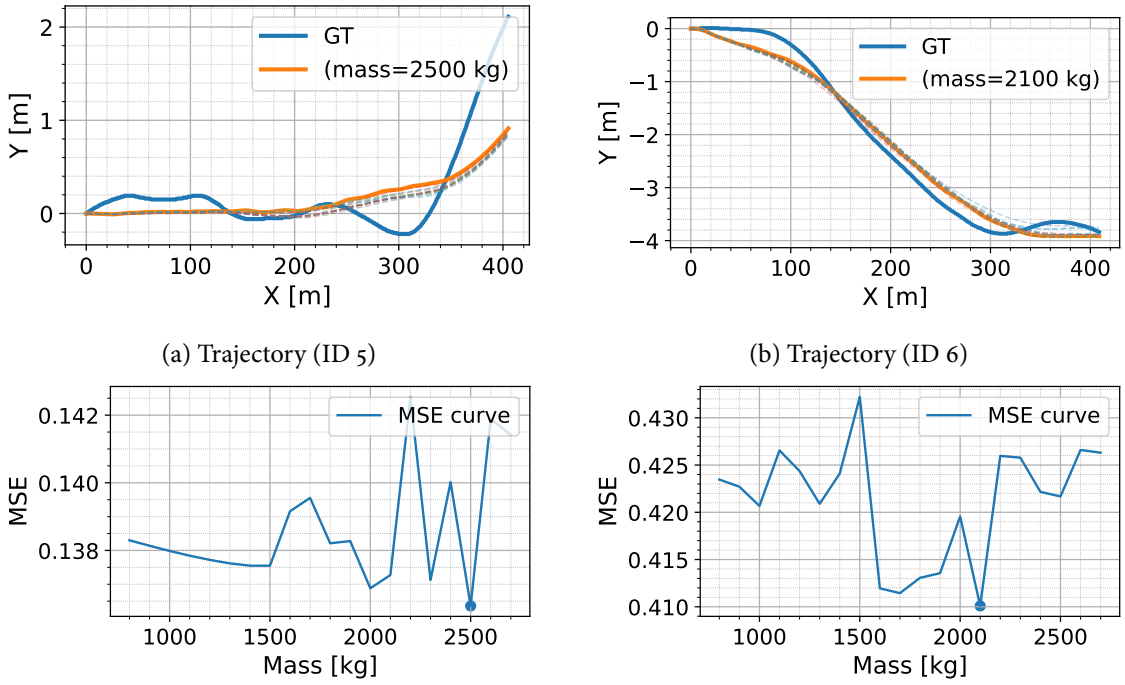
Figure 3.10: Mass estimation results for IDs 3 and 4.

Table 3.6: HighD vehicle mass estimation results

ID	HighD ID	std	MSE	est. mass [kg]
1	50_1329	0.02354	0.00131	1200
2	51_1106	0.02031	0.00120	1500
3	51_1139	0.01636	0.00089	800
4	52_1542	0.03265	0.00225	1200
5	52_696	0.01804	0.00094	1800
6	54_487	0.01117	0.00083	1500
7	54_77	0.02691	0.00154	1000
8	55_2086	0.02205	0.00268	800
9	56_507	0.01206	0.00081	1800
10	50_1332	0.03177	0.00236	800

The quantitative results of the mass estimation are presented in Table 3.6. The table reports the standard deviation (STD) and MSE for each simulation corresponding to the best-matching configuration. The HighD ID denotes the HighD recording segment and the corresponding trajectory identifier of the source vehicle. Corresponding figures illustrating the vehicle trajectories, MSE, and estimated mass values are provided in Figs. 3.14–3.18.

The evaluation results show an overall mass-estimation accuracy of only 20%. Correct mass values were obtained for simulations with IDs 2 and 6, corresponding to 2 out of 10 evaluated cases. Simulations with IDs 3, 8, and 10 resulted in the minimum estimated mass of 800 kg. The highest estimated mass was 1800 kg for simulations with IDs 5 and 9. An estimated mass of 1200 kg was obtained twice, for simulations with IDs 1 and 4. The



(a) Trajectory (ID 5)

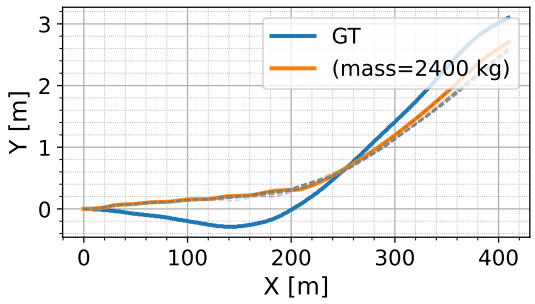
(b) Trajectory (ID 6)

(c) MSE (ID 5)

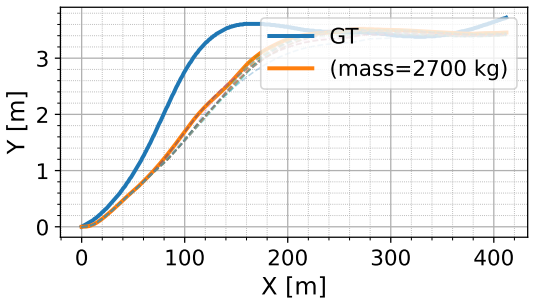
(d) MSE (ID 6)

Figure 3.11: Mass estimation results for IDs 5 and 6.

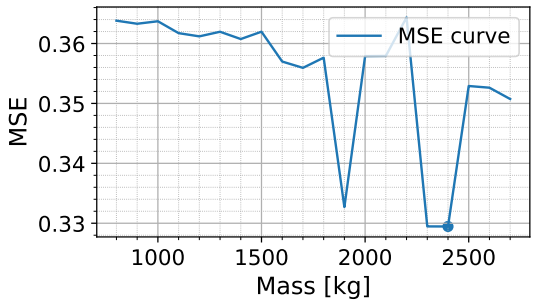
estimation performance is primarily influenced by the limited level of trajectory excitation and the small number of distinct vehicle manoeuvres. All simulations used velocity profiles approximated from the original HighD trajectories. Because these trajectories were recorded on straight highway sections, they predominantly represent steady-state driving and contain few dynamic lateral or longitudinal manoeuvres. Furthermore, the highway trajectories were subsequently smoothed during conversion into logic scenarios, which can suppress small lateral and longitudinal deviations that would otherwise contribute to mass observability. A consequence of this is visible when comparing the MSE and STD values in Tables 3.6 and 3.5, which show a significantly improved trajectory fit around 10 times. This behaviour is expected, as both trajectories in this case are synthetic and generated within the same simulation framework.



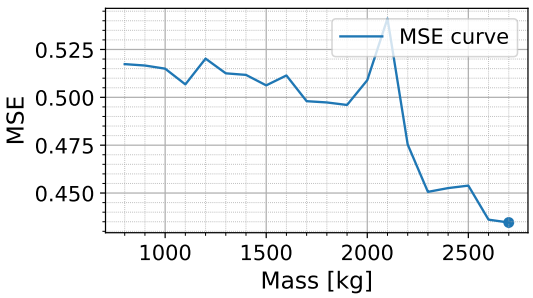
(a) Trajectory (ID 7)



(b) Trajectory (ID 8)

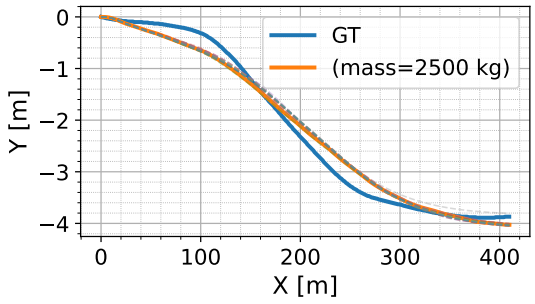


(c) MSE (ID 7)

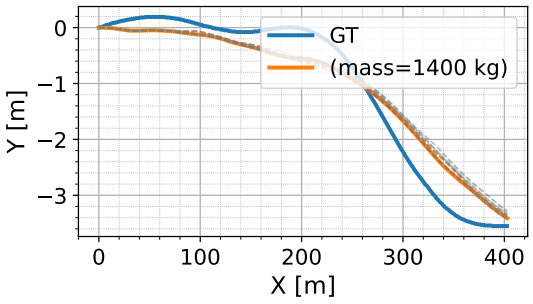


(d) MSE (ID 8)

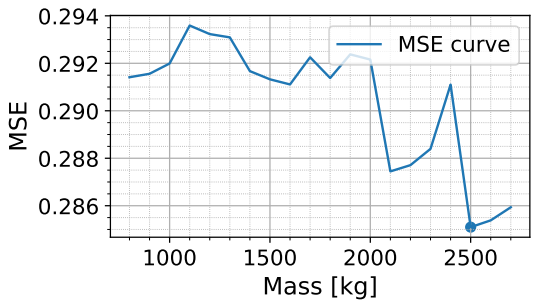
Figure 3.12: Mass estimation results for IDs 7 and 8.



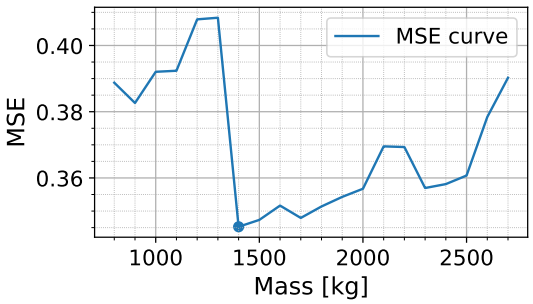
(a) Trajectory (ID 9)



(b) Trajectory (ID 10)

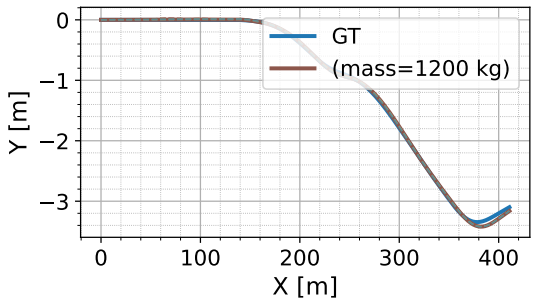


(c) MSE (ID 9)

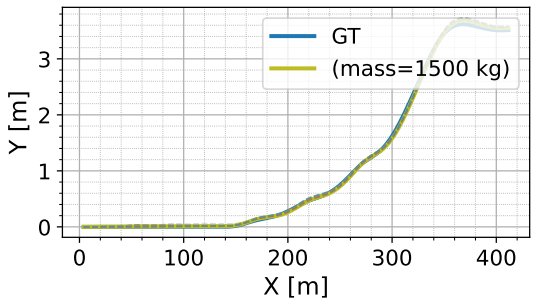


(d) MSE (ID 10)

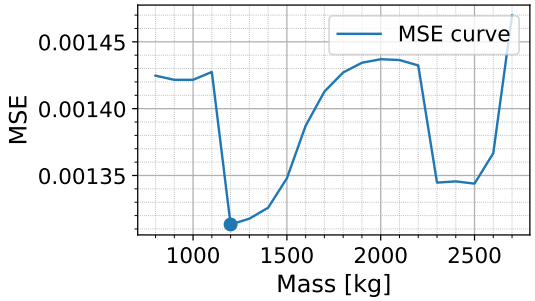
Figure 3.13: Mass estimation results for IDs 9 and 10.



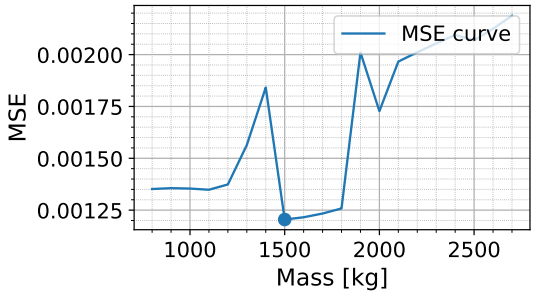
(a) ID 1: HighD 50:1329



(b) ID 2: HighD 51:1106

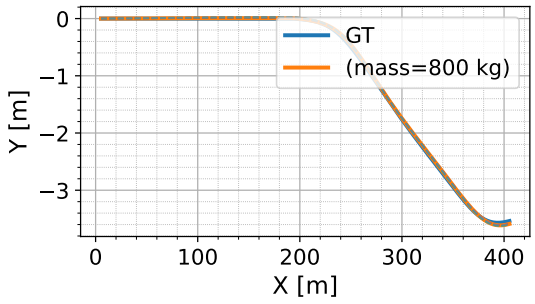


(c) ID 1: MSE

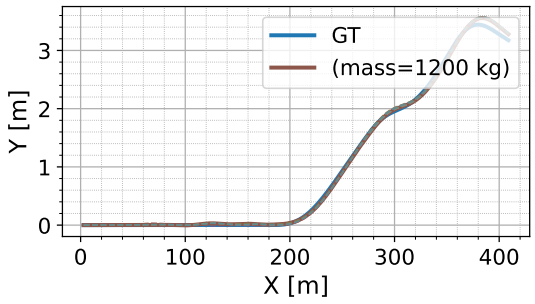


(d) ID 2: MSE

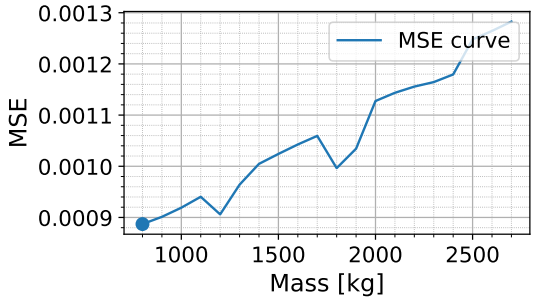
Figure 3.14: Mass estimation results for IDs 1 and 2.



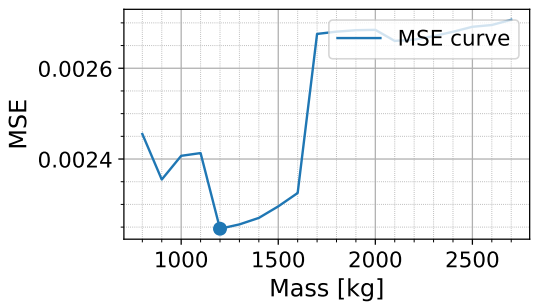
(a) ID 3: HighD 51:1139



(b) ID 4: HighD 52:1542

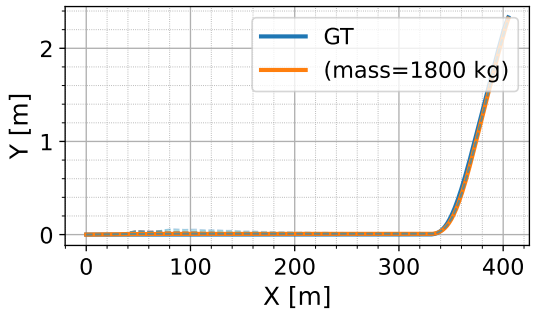


(c) ID 3: MSE

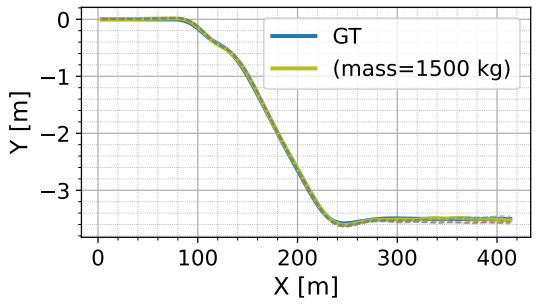


(d) ID 4: MSE

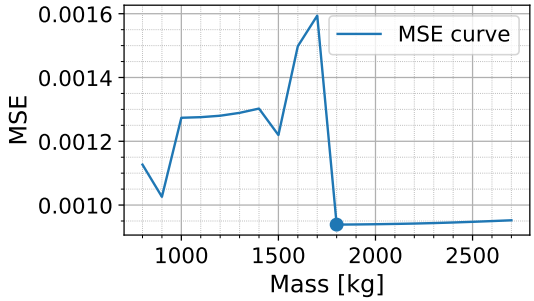
Figure 3.15: Mass estimation results for IDs 3 and 4.



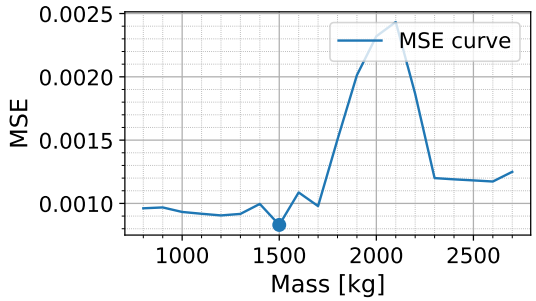
(a) ID 5: HighD 52:696



(b) ID 6: HighD 54:487

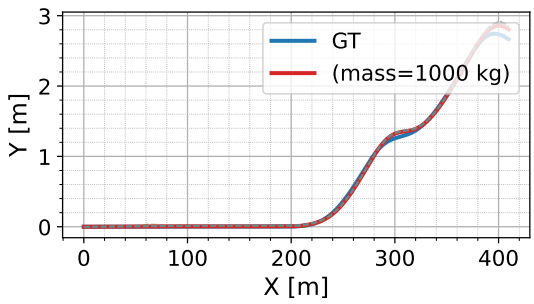


(c) ID 5: MSE

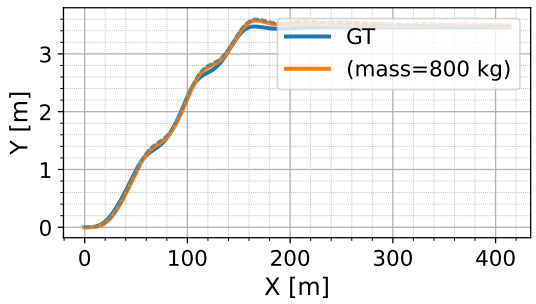


(d) ID 6: MSE

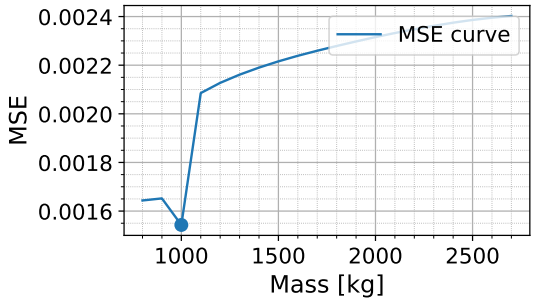
Figure 3.16: Mass estimation results for IDs 5 and 6.



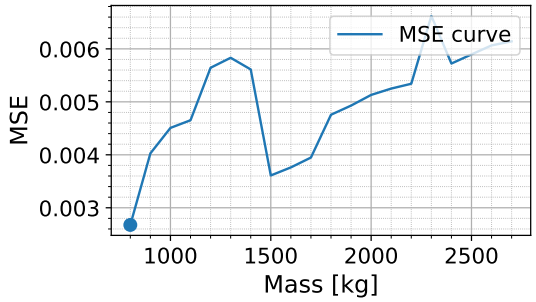
(a) ID 7: HighD 54:77



(b) ID 8: HighD 55:2086

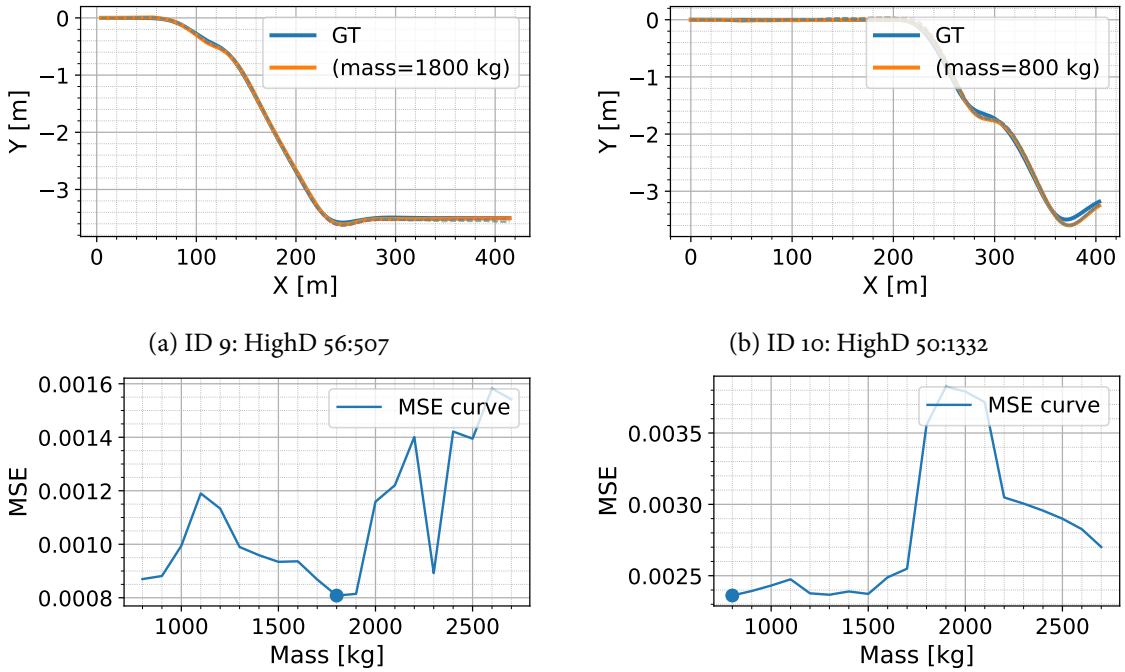


(c) ID 7: MSE



(d) ID 8: MSE

Figure 3.17: Mass estimation results for IDs 7 and 8.



(a) ID 9: HighD 56:507

(b) ID 10: HighD 50:1332

(c) ID 9: MSE

(d) ID 10: MSE

Figure 3.18: Mass estimation results for IDs 9 and 10.

3.4 SUMMARY

This chapter presented an extended-horizon MPC framework for vehicle trajectory filtering, reconstruction, and parameter estimation. Unlike conventional smoothing approaches, the proposed method reconstructs trajectories by driving a simulated vehicle model through measured data, treating the recorded trajectory as a sequence of goal states. This yields a disturbance-reduced trajectory with higher temporal resolution, physically consistent vehicle states, and corresponding control inputs. The method was formulated as a two-stage process. First, missing samples were patched and vehicle states were reconstructed from position-only measurements using basic kinematic relations. Second, a modified MPC formulation with an extended future-state cost term was introduced, enabling anticipation of upcoming goal states while tracking the current target. This modification allows the simulated vehicle to progress smoothly through the trajectory while maintaining consistency between prediction horizon length and remaining goal states. The performance of the proposed filtering approach was evaluated on synthetic single- and double-lane-change trajectories with varying disturbance levels. Results showed that appropriate tuning of the future-look parameter FL, decay factor Z, and prediction horizon leads to substantially improved trajectory reconstruction compared to the RTS smoother, at higher computational cost. The method is therefore best suited for offline preprocessing of vehicle trajectory data. Finally, the applicability of the framework to vehicle parameter estimation was demonstrated through mass estimation experiments. Both real-world trajectories from the HighD dataset and simulated trajectories with known mass were evaluated.

The results indicate that reliable mass estimation is strongly dependent on trajectory excitation and manoeuvre richness, with limited observability for highway driving data dominated by steady-state motion. Despite these limitations, the experiments confirm that the extended-horizon MPC framework can serve as a unified tool for trajectory filtering, reconstruction, and parameter identification.

4

Datasets and Novel Metrics

This chapter presents the datasets used in this thesis, detailing their acquisition, labelling, and augmentation. It focuses on dataset properties and data manipulation, including optimisation of the data for training and evaluation. The chapter also introduces novel metrics for vehicle lateral action (lane-change) classification, motivated by shortcomings of standard measures (e.g., confusion matrices and accuracy) when classification is used for logical scenario reconstruction.

4.1 Introduction

This chapter is divided into two sections. The first section describes the datasets used in this thesis. The overall dataset combines publicly available real-world vehicle trajectory datasets with a custom-generated simulation dataset. The real-world datasets provide realistic traffic context and measurement characteristics, while the simulation dataset provides controllable ground truth labels and supports structured scenario generation.

The second section introduces novel metrics for lane change classification. When classification is used for logical scenario reconstruction, temporal continuity, alignment of the classification centre, and precise detection of action transitions are more important than overall accuracy and confusion-matrix-based metrics. Even 98% accuracy can contain short misclassification gaps that significantly reduce the usability of the reconstructed logical scenarios.

4.2 Datasets for Lane Change Classification

This section covers the datasets used for creating trajectory samples for vehicle lateral classification. It includes the origin of the data, the labelling procedure, and augmentation applied to obtain a diverse and noise-robust dataset.

4.2.1 *HighD*

The *HighD* dataset [67] comprises top-down drone recordings at 6 German motorway sites. Recorded raw video (4K, 25 fps) was processed to extract vehicle trajectories, and trajectory smoothing was applied to reduce measurement noise. In total, the *HighD* dataset includes 60 recordings totalling 16.5 h, about 110,000 tracked vehicles, approximately 45,000 km

travelled, and roughly 5,600 complete lane changes. The dataset provides trajectory time series together with recording metadata.

Road lane lines required for the classification methodology in this thesis were derived from the recording metadata accompanying each *HighD* scene. The metadata provide lateral positions of the upper and lower road lane lines; these were parsed and ordered to obtain candidate line sets for each lane. The resulting constant offset was then applied uniformly to all lines. After processing all recordings, duplicate entries were removed. The positions were finally consolidated into the lower and upper road lane line sets. Since the *HighD* dataset does not provide explicit lane change labels, lane change actions were annotated manually. The manual labelling involved placing markers in approximate areas corresponding to the start and end positions of the lane changes. While this method of data labelling may introduce minor imprecisions, it is acceptable given the non-deterministic nature of the conversion process.

4.2.2 *NGSIM*

The *NGSIM* dataset [68] provides real-world vehicle trajectories extracted from video recordings of highway traffic. Trajectory time series include longitudinal and lateral motion and, therefore, provide suitable input for lane-change related learning tasks. As with *HighD*, lane change labels are not provided and were added manually using the same marker-based annotation procedure.

4.2.3 *HLCsim*

To obtain a large set of trajectories with ground-truth lane-change labels, a custom dataset was created using the dSPACE ASM simulation toolbox [66]. Although there are trajectory-based datasets, such as HighD [67], they lack labels for lane changes and this work can be seen as a step in the development of automated annotation of such real-world data. Since the goal was to generate a large number of diverse scenarios, simulated logic scenarios were run repeatedly while varying parameters in each pass. Base scenarios consisted of multiple lane-keeping and lane-change segments arranged sequentially, as shown in Fig. 4.1. The road was defined as a straight seven-lane highway, with all lanes travelling in the same direction, and curvature was omitted because conversion to the curvilinear coordinate system would neglect it. Vehicle positions were recorded as lateral deviations relative to the road.

The parameters were varied in two ways: either randomly using a uniform distribution or linearly using a preset step size. The parameters varied during the simulation were the duration of each block, lateral and longitudinal velocity, lateral offset, and lane-change type. The data were recorded with a sample interval of 0.02 seconds, after which 1:40 subsampling was applied to reduce the total size of the dataset. This resulted in 2912 driving scenarios covering 1002 km and more than 20 hours of simulated driving, with 975 left lane changes and 1145 right lane changes. The aforementioned dataset is publicly available under the name HLCsim [69].

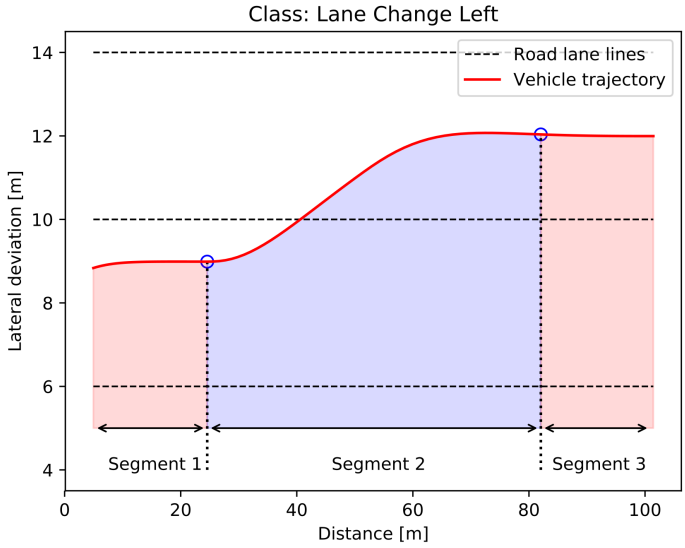


Figure 4.1: Base scenario for dataset generation

4.2.4 *Dataset Composition and Augmentation*

Due to the limited availability of datasets containing annotated lane change actions, a combined dataset was curated from three sources: *HighD* [67], *NGSIM* [68], and *HLCsim* [69]. Since the *HighD* and *NGSIM* datasets do not contain labels for lane change actions, labels had to be added manually. The *HLCsim* dataset contains ground truth lane change labels generated by the simulation. To broaden the dataset and improve robustness to disturbances, it was enhanced through augmentation [70]. Augmentation was applied in two steps. First, trajectories were mirrored from left to right. This mirroring process produces the opposite lane change of the original, increasing class balance between lane change left and lane change right. Second, white noise was applied at different signal-to-noise ratios: 70 dB, 50 dB, and 20 dB. Noise was applied to trajectory signals according to the expressions defined in Subsection 2.7.1. Table 4.1 summarizes the dataset composition and the effect of augmentation steps. Counts are reported per source dataset and per manoeuvre class. Figure 4.2 shows the partitioning applied during dataset preparation to obtain the learning and validation subsets.

The encoding of label classes was defined as follows:

- Lane keep: 0;
- Lane change right: 1:
- Lane change left: 2:

4.2.5 *Synthetic Reference Dataset*

This subsection covers datasets that contain reference logic scenarios used for validating scenario creation workflows and dataset generation pipelines. The dataset is defined as a structured set of parameterised logic scenarios, including systematic parameter variations

Table 4.1: Dataset composition and augmentation

Description	Dataset:	HighD		NGSIM		HLCsim	
	type:	learn	val.	learn	val.	learn	val.
Initial	LK	2266	551	496	121	634	158
	LCL	2353	578	477	118	776	199
	LCR	1939	483	194	55	921	224
After mirroring	LK	4532	551	992	121	1268	158
	LCL	4292	587	671	119	1697	199
	LCR	4292	483	671	55	1697	224
After adding noise	LK	18128	571	3968	121	5072	158
	LCL	17168	578	2684	119	6788	199
	LCR	17168	483	2684	55	6788	224
Sum		52464	1632	9336	295	18648	581
Learn/valid [%]		65.21	65.07	11.61	11.76	23.18	23.17
Total num. of traj.		80448	2508				

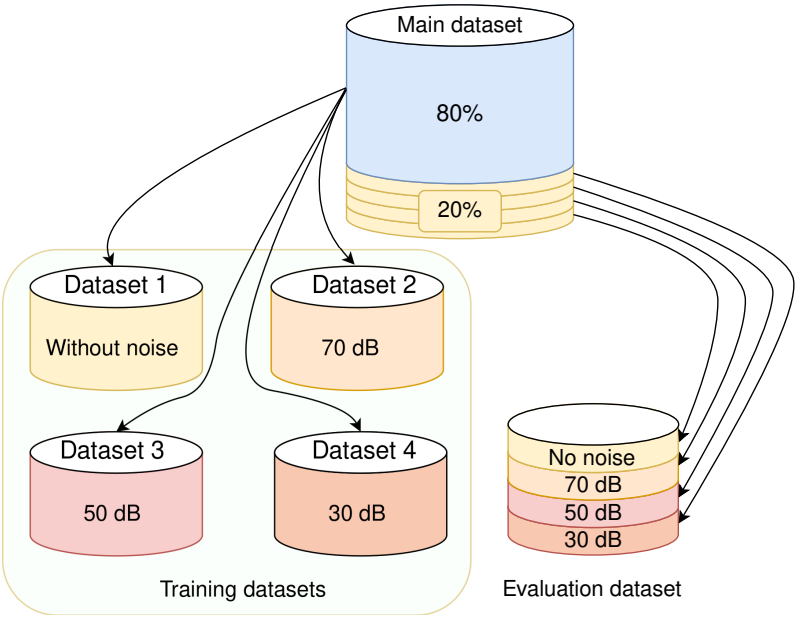
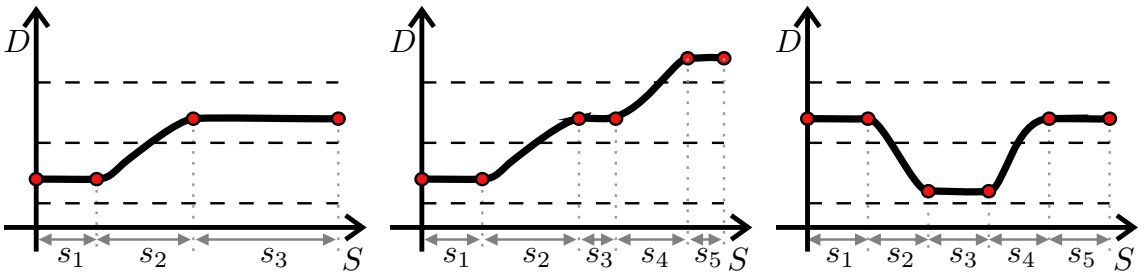


Figure 4.2: Dataset partitioning into learning and validation subsets

and controlled distortions applied to the generated trajectories. A reference scenario dataset was created starting from three base logic scenarios: *SingleLC*, *DoubleLC*, and *UshapeLC*, shown in Fig. 4.3. Each scenario consists of a sequence of segments with configurable longitudinal acceleration, lateral velocity, and segment duration. Multiple test cases (TC) were defined by varying these parameters, as listed in Table 4.2. In addition, multiple distortion configurations (DC) were defined to systematically disturb the generated trajectories, as listed in Table 4.3. The complete dataset, consisting of all combinations of base scenarios, test cases, and distortion configurations, is publicly available under the name *LC Scenarios for Lane Change Classification* [71].



(a) SingleLC (b) DoubleLC (c) UshapeLC

Figure 4.3: Reference lane change logic scenarios.

Table 4.2: Test case parameter configurations for reference logic scenarios

Scenario name:			Single LC				Double LC				U-shape LC			
segment	parm.	unit	TC1	TC2	TC3	TC4	TC1	TC2	TC3	TC4	TC1	TC2	TC3	TC4
s_1	a	[m/s^2]	0	0	0.2	0.1	0	0	0.2	0.2	0	0	0.2	0.2
	v_{lat}	[m/s]	-	-	-	-	-	-	-	-	-	-	-	-
	t	[s]	10	10	10	10	10	10	10	10	10	10	10	10
s_2	a	[m/s^2]	0	0	0	0.1	0	0	0	0.2	0	0	0	0.2
	v_{lat}	[m/s]	1	2	1	1	1	1	1	1	1	1	1	1
	t	[s]	8	8	8	8	8	8	8	8	8	8	8	8
s_3	a	[m/s^2]	0	0	-0.2	0.1	0	0	0.2	-0.2	0	0	0.2	-0.2
	v_{lat}	[m/s]	-	-	-	-	-	-	-	-	-	-	-	-
	t	[s]	5	5	5	5	6	6	6	6	6	6	6	6
s_4	a	[m/s^2]	-	-	-	-	0	0	0	0	0	0	0	0
	v_{lat}	[m/s]	-	-	-	-	1	2	1	2	1	2	1	2
	t	[s]	-	-	-	-	8	8	8	8	8	8	8	8
s_5	a	[m/s^2]	-	-	-	-	0	0	-0.2	0.2	0	0	-0.2	0.2
	v_{lat}	[m/s]	-	-	-	-	-	-	-	-	-	-	-	-
	t	[s]	-	-	-	-	5	5	5	5	5	5	5	5

4.3 NOVEL METRICS

For vehicle behaviour classification, authors such as Sonka et al. [72] used the confusion matrix, while Schlechtriemen et al. [73] and Zekany et al. [74] used the true positive rate and false positive rate. The works in [75, 76, 77] used only accuracy. Zhang et al. [78] used both the confusion matrix and accuracy to interpret their results. However, due to the specific way in which the output is used, accuracy and the confusion matrix alone are not sufficient to fully describe classification quality. To improve the quality assessment of the classification output, three additional metrics are proposed: *Centre Offset*, *Start/End*, and *Estimate Usability*.

Table 4.3: Distortion configurations applied to generated trajectories

Distortion ID	NSR [dB]	r_p[%]	O_p[%]	O_s[]
DC1	w/n	0	0	0
DC2	50	0	0	0
DC3	30	0	0	0
DC4	25	0	0	0
DC5	w/n	5	0	0
DC6	w/n	0	1	1
DC7	w/n	5	1	1
DC8	50	5	1	1

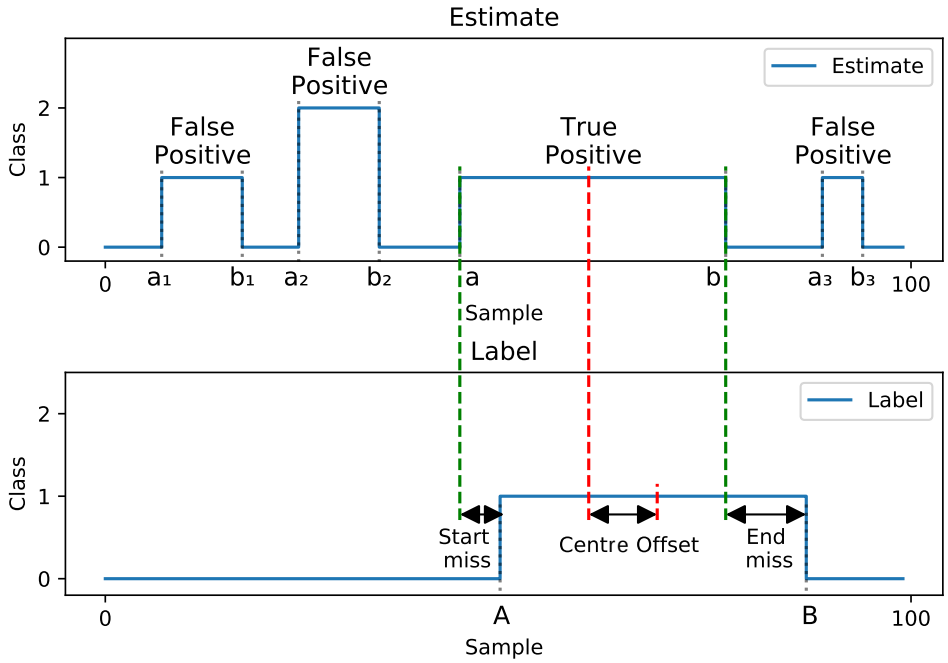


Figure 4.4: Parameters for classification assessment

4.3.1 *Centre Offset*

The first metric, referred to as the *Centre Offset*, represents the difference in the positions of the centres of the lane change in the label and the estimate. When a logic simulation scenario is generated from classification output with a low *Centre Offset* value, positional precision during the lane change may be reduced. The *Centre Offset* is calculated as a percentage by computing the distance error between the centre of the labelled lane change C_{AB} and the centre of the estimated lane change C_{ab}, divided by the number of samples in the scenario, N.

$$\text{Centre offset} = \frac{|C_{AB} - C_{ab}|}{N}. \tag{4.1}$$

Fig. 4.4 illustrates how the centre points of lane changes are defined for metric calculation. Example 3 in Fig. 4.5 shows a case where the label and estimate differ in their centre positions.

Table 4.4: Classification assessment on given examples

no.	Acc.	TP	TN	FP	FN	Start/ End	Centre offset	Estimate usability
1.	0.9	0.3	0.633	0.033	0.033	0.9	1	1
2.	0.9	0.3	0.633	0.033	0.033	0.9	1	1
3.	0.85	0.283	0.617	0.05	0.05	0.95	0.925	1
4.	0.65	0.217	0.55	0.117	0.117	0.95	0.875	0.778
5.	0.97	0.323	0.657	0.01	0.01	0.77	0.965	0.259
6.	0.64	0.21	0.547	0.12	0.123	0.83	0.835	0.5

The results in Table 4.4 demonstrate that such offsets reduce alignment even when other metrics remain high.

4.3.2 *Start/End Metric*

The *Start/End* metric represents the extent to which the label and estimate differ in their start and end positions. It quantifies the discrepancy between the start and end points of the label and the estimate and can be calculated with the following expression:

$$\text{Start/End} = 1 - \frac{|B - A + b - a|}{N}. \tag{4.2}$$

A higher Start/End value indicates a better match between the label and estimate in terms of their respective starting and ending positions, while a lower value indicates greater deviation. The metric has a maximum value of 1, which represents a perfect match between the starting and ending positions of the estimate with the label. A logic simulation scenario based on classification output with a low Start/End value may start the lane change too early or too late. Examples 1 and 2 in Fig. 4.5 demonstrate such early and delayed detections. As shown in Table 4.4, these two cases yield identical metric values due to their symmetric nature: an underestimate in Example 1 versus an overestimate in Example 2.

4.3.3 *Estimate Usability*

The third metric, *Estimate Usability*, quantifies the extent to which the estimate is "usable" for the reconstruction of logic scenario segments. It is defined as the ratio of false positives to the sum of all positives, as given by the following expression:

$$\text{Estimate usability} = \frac{b - a}{\sum_{i=1}^{n} (b_i - a_i) + (b - a)}. \tag{4.3}$$

A lower value of *Estimate Usability* indicates that some of the generated lane change segments in the simulation scenario may not be present in the input data. To better understand the metrics and their respective formulae, refer to Fig. 4.4. The lower part of Fig. 4.4 displays a labelled left lane-change action, while the upper part illustrates an estimated detection example. Fig. 4.4 shows the parameters necessary for calculating the proposed metrics, including *Centre Offset*, *Start/End* errors, and false positive detections of both the same and opposite types of lane changes. Examples 4–6 in Fig. 4.5 illustrate the impact

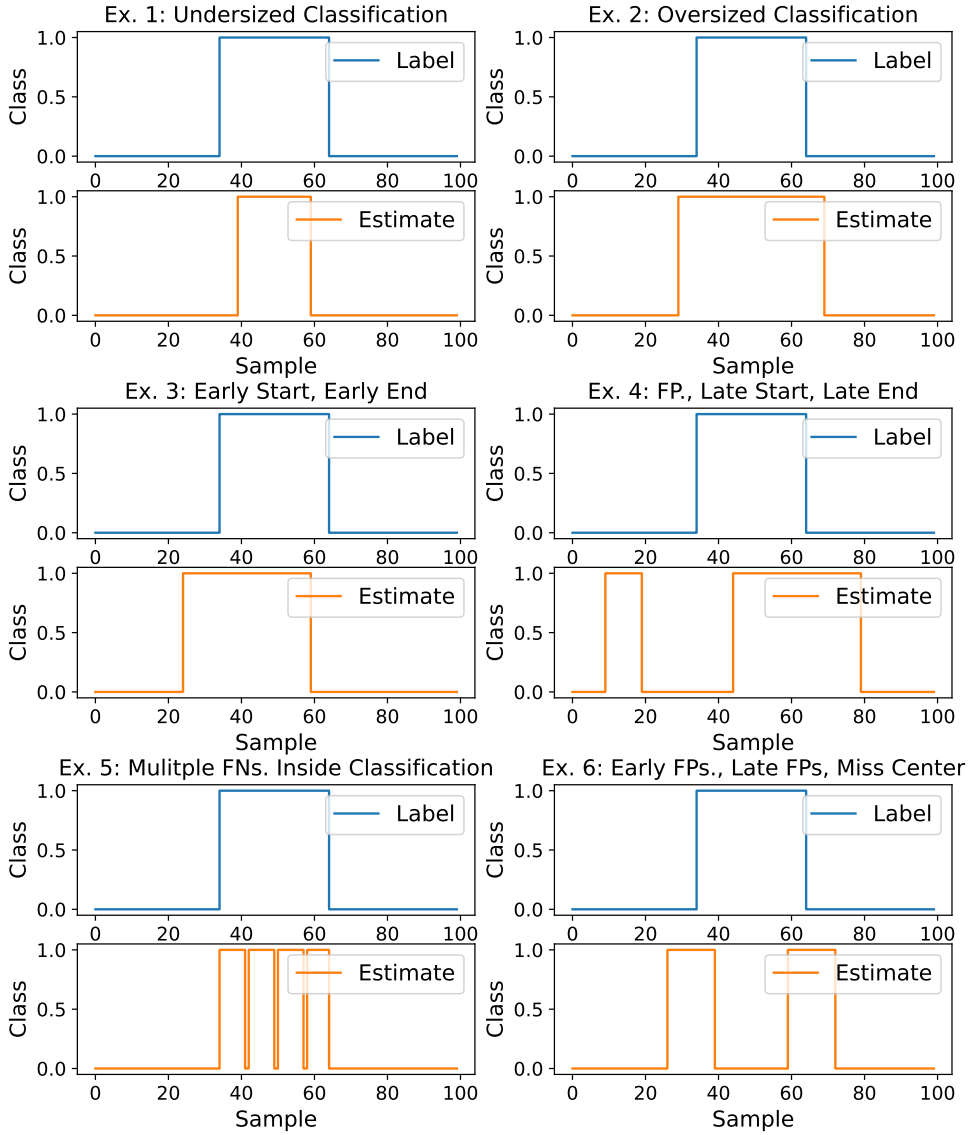


Figure 4.5: Classification examples

of false detections on usability. Example 4 shows that even when Start/End alignment remains high, false positives significantly reduce the metric. Example 5 demonstrates how multiple detections, where only one is expected, can lead to unrealistic lane assignments, while Example 6 highlights that severe false detections further decrease usability. The corresponding metric values are reported in Table 4.4.

4.4 SUMMARY

This chapter presented the datasets used in this thesis, including real-world vehicle trajectory datasets (*HighD* and *NGSIM*), a synthetic simulation dataset with ground truth manoeuvre labels (*HLCsim*), and a structured reference scenario dataset (*LC Scenarios for Lane Change Classification*). The acquisition, labelling, and augmentation procedures described here establish the data foundation for the subsequent methodology and experimental analyses. The chapter also introduced three novel metrics for evaluating lateral vehicle action classification in the context of logic scenario reconstruction. In contrast to accuracy and confusion-matrix-based measures, the proposed metrics explicitly assess temporal consistency and segment quality, which are essential when classification outputs are used to generate logic scenario segments. The *Centre Offset* metric measures the displacement between the centres of the labelled and estimated lane change segments, describing manoeuvre alignment within the trajectory. The *Start/End* metric quantifies the agreement between estimated and labelled segment boundaries, capturing early or delayed detections that alter segment duration. The *Estimate Usability* metric evaluates practical usability for scenario reconstruction by penalizing false positive lane change detections that would introduce non-existent manoeuvres into the generated logic scenario. Representative examples illustrated that high accuracy can still correspond to poor segment quality due to short misclassification gaps, shifted centres, or spurious detections. Together, these metrics provide a more application-relevant assessment of lane change classification performance and support a fairer comparison of methods when the downstream objective is robust logic scenario generation.

5

Vehicle Action Classification and Logic Segments

This chapter presents the methodology for classifying vehicle actions. It further details how the resulting classifications are systematically transformed into logic scenarios. Vehicle action classification is divided into lateral and longitudinal components. Lateral classification is performed using two approaches, polynomial modelling and neural networks, while longitudinal classification is performed through greedy knot reduction. The chapter concludes with a comprehensive evaluation of the complete process. Logic segments are first constructed from the combined lateral and longitudinal classifications and subsequently used to form logic scenarios. These scenarios are then simulated and evaluated using both synthetic and publicly available datasets.

5.1 Introduction

Vehicle action classification aims to identify the type of action being performed and to accurately determine its start and end points within the trajectory. Accurate classification enables the conversion of raw trajectory data into structured logical descriptions that can be directly utilised in simulation environments. Within the scope of this thesis, vehicle action classification is divided into longitudinal and lateral categories. Longitudinal actions are segmented by constant acceleration values into three classes: accelerating, decelerating, and constant velocity. Lateral actions are classified into three categories: lane change left, lane change right, and lane keeping. This classification scheme was chosen to ensure compatibility with widely used simulation software, such as dSPACE ASM ModelDesk [66], IPG CarMaker [79], CARLA [80], and others. The longitudinal classification of vehicle actions using constant-acceleration segments is illustrated in Fig. 5.1, while the classification of vehicle lateral actions is illustrated in Fig. 5.2.

The requirements for vehicle action classification begin with representing the vehicle trajectory in the curvilinear coordinate system (S, D), described in Section 2.3, along with the time vector T. Normalisation is applied only for lateral classification, serving as a simplification for the neural-network-based method. Successful classification requires more than just the vehicle trajectory; it also depends on road geometry and lane line information, which provide the reference against which lateral actions are defined. Road lane lines R are likewise represented as trajectories in the curvilinear coordinate system. All required vectors and their sizes are given in Table 5.1. The variable n denotes the number of samples, and for road lane lines the dimension m denotes the number of lane lines.

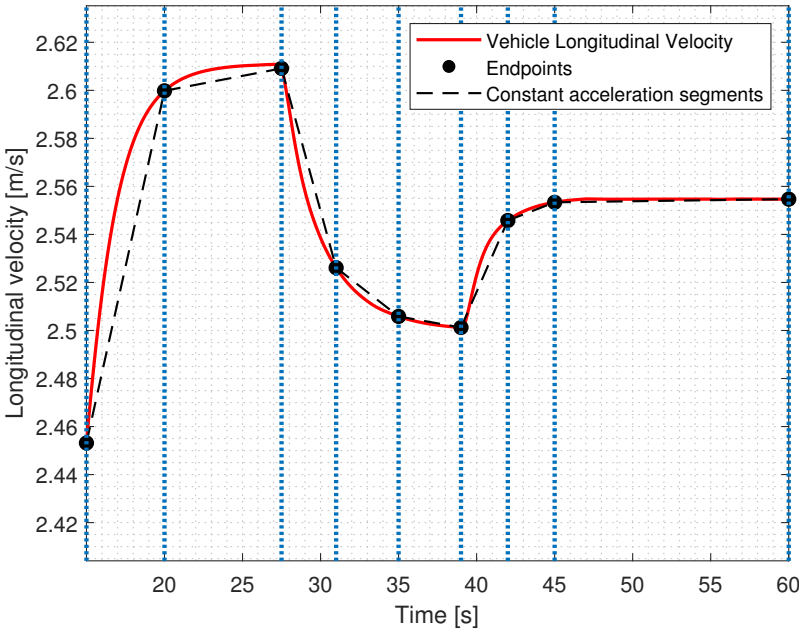


Figure 5.1: Longitudinal classification example

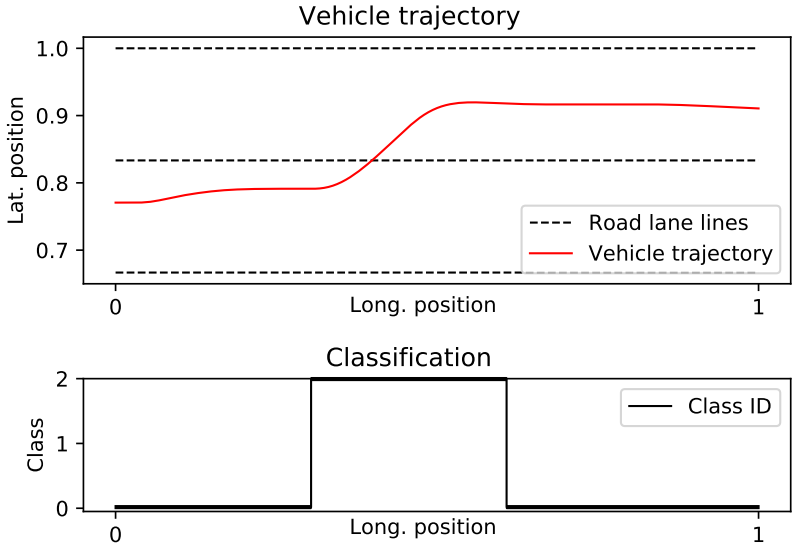


Figure 5.2: Lateral classification example

This chapter continues with Subsections 5.2.1 and 5.2.2, which present polynomial modelling and neural network methods for lateral classification, respectively. Section 5.3 discusses the classification of longitudinal vehicle actions. Finally, Subsection 5.4.2 addresses the integration of longitudinal and lateral actions into combined logic segments, while Section 5.5 provides the performance evaluation of the complete conversion process.

5.2 Classification of Lateral Vehicle Actions

This section addresses the classification of vehicle lateral actions using two complementary methods. The first method relies on polynomial modelling, where lane crossings are detected and the corresponding start and end points are approximated from the inflection points of a

Table 5.1: Requirements for vehicle action classification

Mark	Size	Description
S	$1 \times n$	Longitudinal trajectory in curvilinear coordinates.
D	$1 \times n$	Lateral trajectory in curvilinear coordinates.
T	$1 \times n$	Time vector corresponding to the trajectory samples.
R	$m \times n$	Road lane lines represented as trajectories in curvilinear coordinates

third-degree polynomial. This method is inspired by Nelson et al. [81], who modelled vehicle movement with a polynomial, and Zhou et al. [82], who modelled vehicle lane changes using a hyperbolic tangent function and compared them with a quintic polynomial model. The second method employs a custom neural network architecture, first presented in the author's earlier work [83] and later extended in [7]. This neural network method was chosen due to the well-established effectiveness of neural networks in classification tasks, which are widely adopted in this field [75, 84, 78, 85]. Beyond neural networks, alternative strategies have also been explored. Hülnhagen et al. [77] applied fuzzy logic, while Lin et al. [86] proposed a hybrid framework that integrates neural networks with fuzzy logic. Similarly, Sonka et al. [72] investigated probabilistic and fuzzy classification methods, reporting improvements in classification robustness. To benchmark its performance, the proposed model was compared against several state-of-the-art approaches from the literature. Ashfaq et al. (2021) [87] introduced the Dual Attention Bi-LSTM model, which combines bidirectional LSTMs with dual attention mechanisms to enhance temporal and contextual feature extraction for lane change prediction. Zhou et al. (2023) [88] proposed a multi-task learning framework (LK-/LC-LSTM) that jointly classifies lane-keeping and lane-change actions, improving accuracy by exploiting shared representations across related tasks. Liu et al. (2022) [89] developed a sequence-to-sequence (TTS) deep learning model that leverages temporal dependencies in surrounding vehicle trajectories to predict lane change intentions in real-time. Both methods use the same labelling scheme, classifying actions into three classes: lane keeping with value 0, lane change left with value 2, and lane change right with value 1.

5.2.1 *Polynomial Lane Change Classification Method*

The polynomial method is an analytical approach for classifying lateral vehicle movements in trajectories represented in the curvilinear coordinate system, as described in Section 2.3. This approach offers an alternative to the most common classification solution, namely neural networks. In the polynomial method, lane change classification involves fitting a third-degree polynomial around the point where the vehicle crosses the lane line. The polynomial method consists of two phases: (i) identifying lane line crossings and (ii) determining the start and end points of a lane change.

To unambiguously find lane line crossing points, despite noise and other unwanted disturbances in the data, lane line crossings are calculated as the intersection of two polynomials. It is well-known that complete vehicle trajectories and road lane lines cannot be accurately represented using a single polynomial of any degree. Thus, the solution is to create a family of polynomials that approximate the trajectory or the road lane lines, where each family member represents only a segment of the input data. Such a solution with

families of polynomials can be seen in Figure 5.3. The criterion function for determining the size of each segment for a specific polynomial is calculated using the following formula:

$$J = \frac{\frac{1}{n}\sum_{i=1}^{n}(Y_{poly} - Y_{input})^2}{\log_e(\frac{n}{2})}. \tag{5.1}$$

Lane line crossings are detected by comparing each lane line polynomial family to the

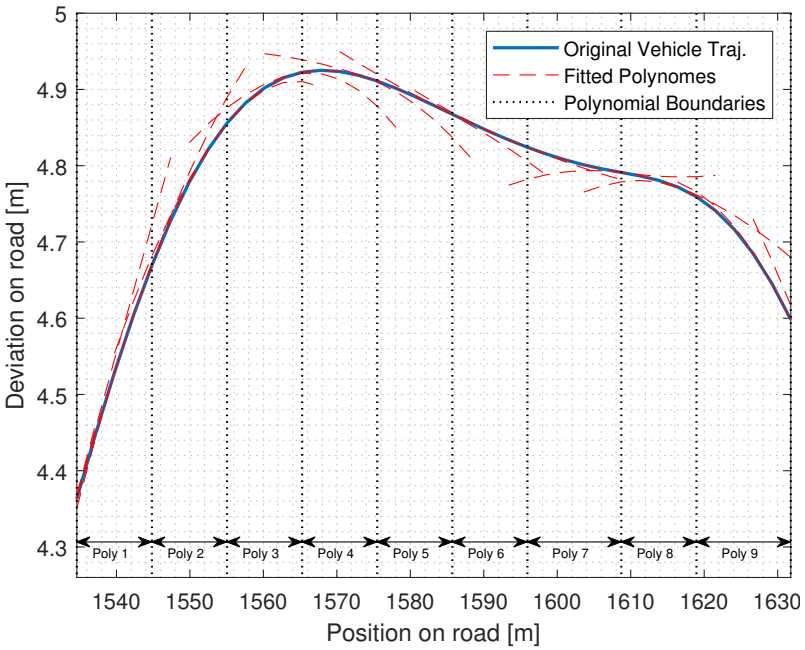


Figure 5.3: Vehicle trajectory conversion to a family of polynomials

vehicle trajectory family and checking whether the two polynomials intersect. This approach allows the detection of lane changes with subsample precision, since both road lane lines and the vehicle trajectory are converted to a continuous representation. Fig. 5.4 illustrates a simplified example of the detection process. The dashed lines represent the fitted polynomials for both the vehicle trajectory and the road lane lines. The range of each polynomial fit is indicated by black arrows. In this example, a lane-line crossing is detected at the intersection of the polynomial parts for the road marked "Road Poly 2" and the vehicle trajectory marked "Traj Poly 3." Even though this process can be computationally demanding, this is not a limitation, as the algorithm application is strictly run offline. Processing a 15-minute driving scenario can take up to 60 minutes, depending on the number of road lanes and vehicles involved. As mentioned earlier, the advantage of this approach is its ability to work with subsample precision, enabling the detection of lane line crossings even when there are no overlapping samples, as shown in Fig. 5.5. The output in this phase is a list of indices that correspond to lane line crossings in the data.

The next step involves determining the start and end points of the lane change. This is achieved by fitting a third-degree polynomial lane-change model around the detected lane-line crossing, as illustrated in Fig. 5.6. However, this fitting process is not straightforward, as the fitting range starts around the crossing point in the centre and iteratively expands outwards. The range expansion is limited by parameters k_{max} and j_{max}, which represent the

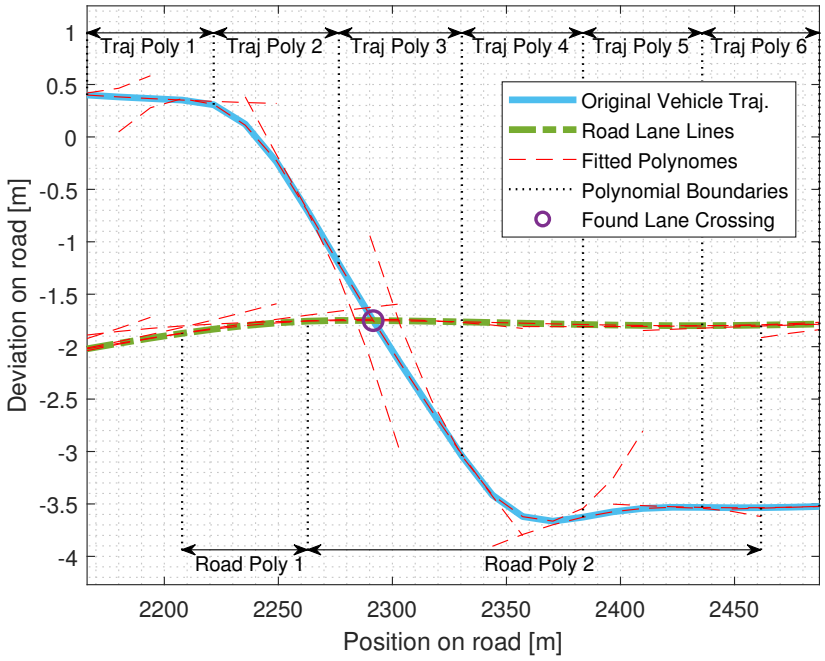


Figure 5.4: Trajectory lane line crossing position detection.

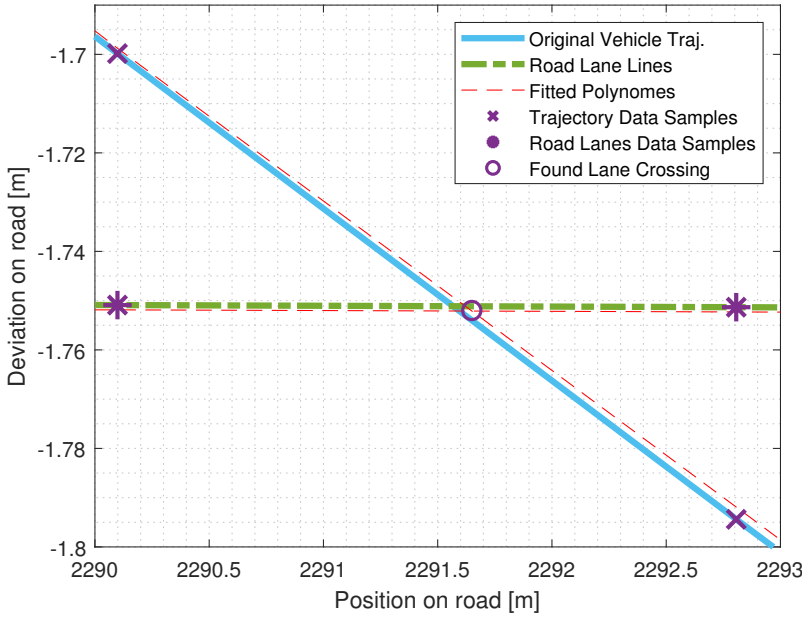


Figure 5.5: Close up on the trajectory lane line crossing position detection.

midpoints between the current lane line crossing point I_k and an earlier I_{k-1} or the next I_{k+1} one, respectively. Parameters k_{max} and j_{max} are calculated using the following formulas:

$$j_{max} = \frac{I_k - I_{k-1}}{2}, \tag{5.2}$$

$$k_{max} = \frac{I_{k+1} - I_k}{2}. \tag{5.3}$$

The range is iteratively extended by parameters j and k, which are limited by j_{max} and k_{max}, and continue expanding until both reach their limits. If one limit is reached earlier than the other, the remaining parameter continues increasing in the following iterations. After the

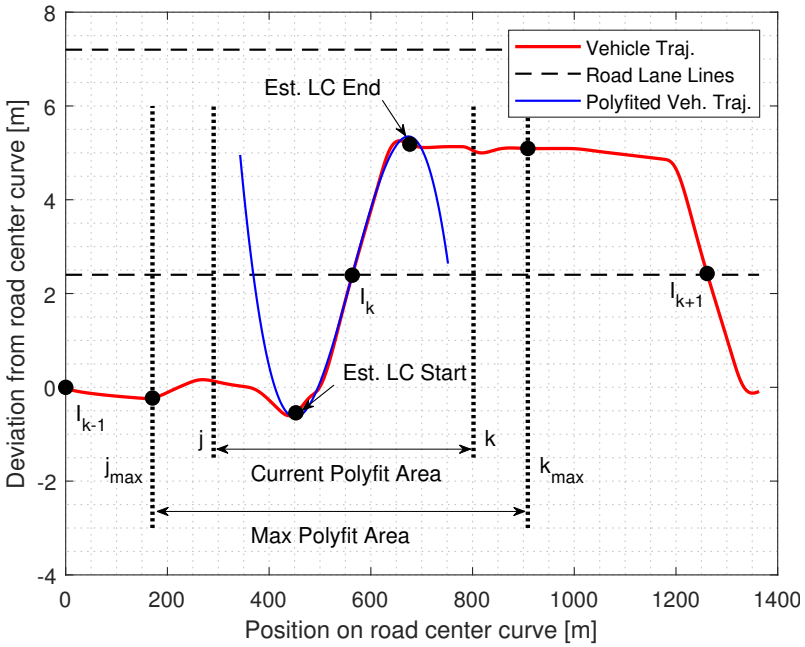


Figure 5.6: Estimation of lane change starting and end points.

current area is fitted by a third-degree polynomial function (5.4), the obtained parameters are used to calculate the inflection points ($LCIs$) (5.5):

$$f(x) = a_3x^3 + a_2x^2 + a_1x^1 + a_0, \tag{5.4}$$

$$LCI_{1,2} = \frac{-a_2 \pm \sqrt{a_2^2 - 3a_3a_1}}{3a_3}, \tag{5.5}$$

where $LCI_{1,2}$ must satisfy:

$$\forall LCI_{1,2} \in: \quad \mathbb{R} \bigcap \langle j_{max}, k_{max} \rangle. \tag{5.6}$$

The subsequent step involves proceeding and computing the value of the criterion function.

$$J_i = \frac{\frac{1}{k+j}\sum_{i=1}^{k+j}(Y_{3rdpoly} - Y_{Poly})^2}{\log_e(k+j)}, \tag{5.7}$$

This value is then compared to the previous one (initially set to $+\infty$), and the algorithm continues until convergence.

The type of a lane change can be determined by comparing the values of the vehicle trajectory at $LCI_{1,2}$.

$$LCT = Y(\min(LCI_{1,2})) - Y(\max(LCI_{1,2})) \tag{5.8}$$

$$\text{sign}(LCT) = \begin{cases} 1: & \text{Lane Change Right} \\ -1: & \text{Lane Change Left} \end{cases} \tag{5.9}$$

Lane change start points are always the smaller root solutions $\min(LCI_{1,2})$, as can be seen by examining (5.5), while the end points are the larger root solutions $\max(LCI_{1,2})$. Apart from performing classification, the estimated polynomial parameters a_3, a_2, a_1 and a_0 can also be used to compute the velocity and acceleration of the lane change.

5.2.2 *Neural Network Lane Change Classification Method*

The neural network approach is proposed as a comparative method, contrasting it with the polynomial method. The optimal neural network model was developed in two stages: first, by experimenting with different types of layers and their configurations [83], and later by comparing it with models proposed by other authors for this specific classification task [7]. Motivated by previous research on vehicle motion classification using neural networks [90], five neural network architectures are introduced, differing in structural complexity as well as in the types of layers employed. This was done to determine which model and layer configuration is most suitable for this specific classification task. The tested models range from simple FC and CNNs, to combinations of convolutional and LSTM network layers, and in the following each of them is presented.

Model 1: Simple FC. A FC network [91] is an introductory network for simple classification problems. The proposed network model is composed of 4 fully connected layers and a softmax layer, where FC layers are composed of the following neuron numbers per layer (256,128,64,32) as shown in Fig. 5.7. The main advantage of the FC neural networks is that no special assumptions about the structure of the input data is needed.

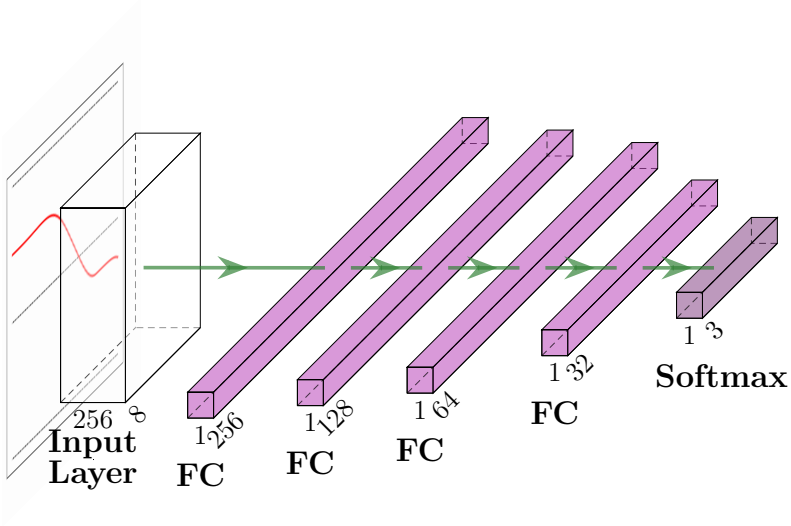


Figure 5.7: Model 1: Simple FC for vehicle motion classification.

Model 2: Simple CNN. This network model is proposed to account for the spatial nature of the input data, as they represent vehicle position relative to road lane lines. The model is composed of three convolutional layers each containing 32 filters and kernel dimensions of 7, 5 and 3, respectively. Convolutional layers are followed by a FC layer with 64 neurons, and a softmax layer for output as shown in Fig. 5.8. The advantage of convolutional layers, which are inspired by visual perception, is that they take tensors as input data and can understand spatial relations (correlation between nearby pixels in the image) [92].

Model 3: CNN + LSTM. Models that combine CNN and LSTM layers were initially developed for visual time series prediction and the purpose of generating textual descriptions from sequences of images [93]. A network model based on a combination of convolutional

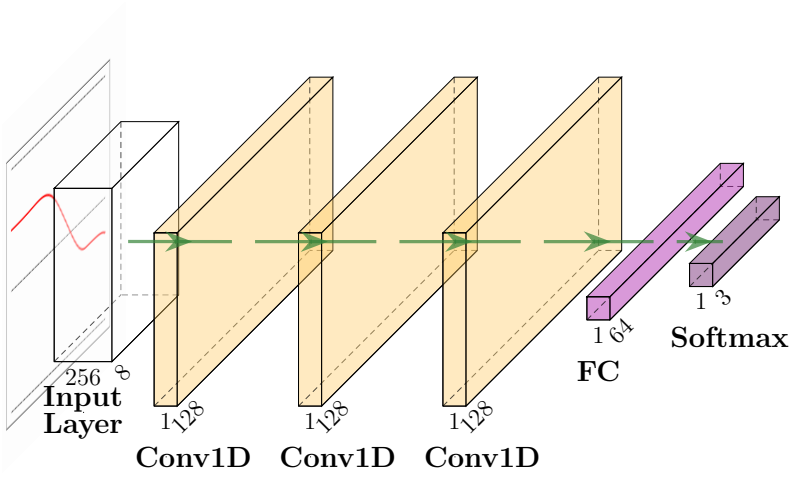


Figure 5.8: Model 2: Simple CNN for vehicle motion classification.

layers with LSTM layers is proposed because of its ability to tackle spatio-temporal information in input data. Configuration of a simple CNN model is extended with an LSTM layer with 32 neurons inserted after 3 convolutional layers and before the FC layer, and the output softmax layer, as shown in Fig. 5.9.

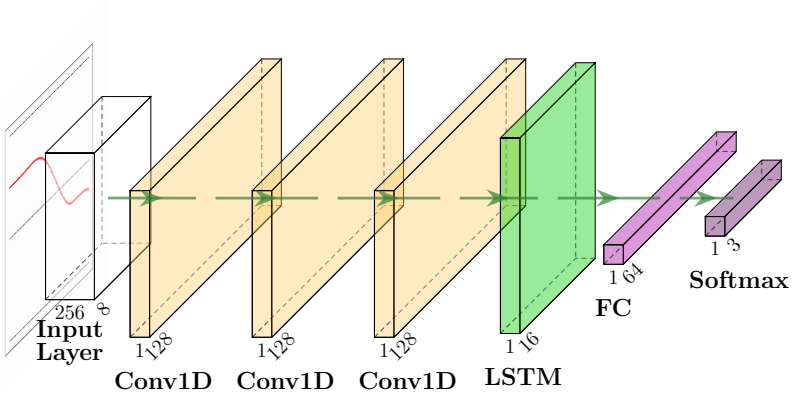


Figure 5.9: Model 3: CNN + LSTM

Model 4: Simple Bi-LSTM. This model was chosen to see how a temporal model deals with structured input data and how structured input data will influence the model's classification capability [78]. Composition of the network is based on two bidirectional LSTM (Bi-LSTM) layers with 64 cells, accompanied by a FC layer with 64 neurons and a softmax layer as output, as shown in Fig. 5.10. The model based on Bi-LSTM is capable of running input data in both directions, which allows it to preserve information from both past and future.

Model 5: Bi-LSTM + MSCNN. This network model was taken from [78], where it was applied to driver-behaviour recognition and achieved an accuracy of 85.27%. According to [78], the advantage of the Bi-LSTM over LSTM in this application is: "Compared to vanilla

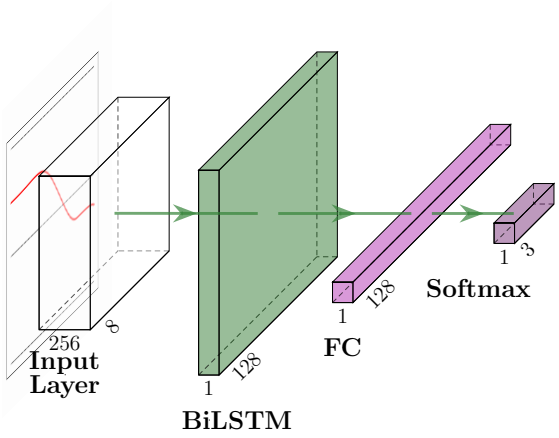


Figure 5.10: Model 4: Simple Bi-LSTM

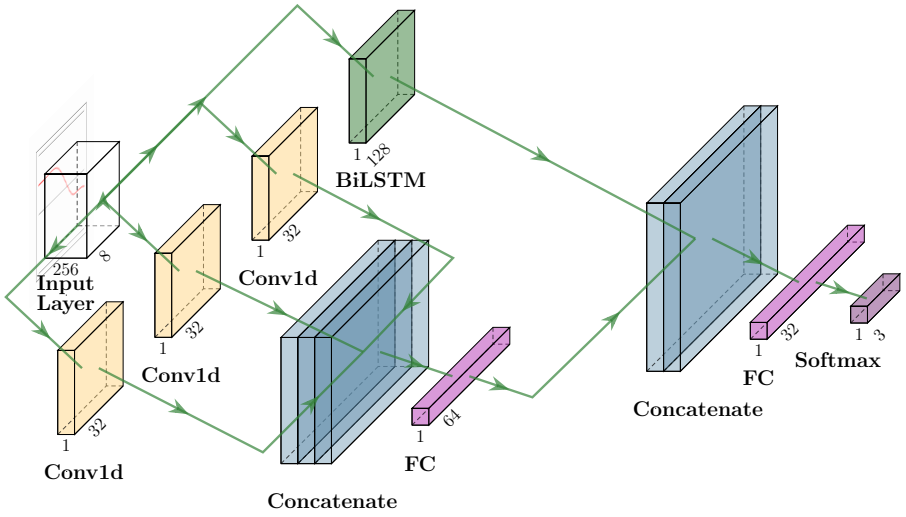


Figure 5.11: Model 5: Bi-LSTM + MSCNN

LSTM, bidirectional LSTM (Bi-LSTM) architecture can use the bidirectional information of the trajectory which prompts us to introduce a Bi-LSTM module to model the dynamic evolution of trajectories." The structure of the model is composed of two main branches which are a Bi-LSTM and multi-scale CNN (MSCNN) followed by a FC layer with 32 neurons, and a softmax output layer. The Bi-LSTM module is composed of two LSTM layers each containing 64 cells, while the MSCNN branch has three parallel convolutional layers, accompanied by a FC layer with 64 neurons. Convolutional layers each consist of 32 filters and kernel sizes of 7,5 and 3, respectively. The combined and augmented dataset used for training and validation of the earlier described neural network models is described in Section 4.2.

Adam [94] was used as the training optimizer based on stochastic gradient descent since it is well suited for problems that are both large in data and the number of parameters. The following metrics for evaluation were used: accuracy calculated as frequency of prediction matching labels, categorical cross-entropy for the loss function, and confusion matrix macro parameters: True positive (TP), True negative (TN), False positive (FP) and False negative

Table 5.2: Evaluation results of the analysed network models on the custom lane change detection dataset

Model Name	Value	Without Noise Epoch=30000	Noise 70 dB Epoch=40000	Noise 50 dB Epoch=50000	Noise 30 dB Epoch=60000
Simple FC	Accuracy	74.4889%	74.0898%	74.6113%	74.3476%
	Loss	0.6881	0.6923	0.6911	0.6952
	TP	0.44	0.44	0.44	0.4333
	TN	1.4366	1.44	1.4399	1.4333
	FP	0.5566	0.56	0,56	0.5667
	FN	0.5566	0.56	0,56	0.5667
Simple CNN	Accuracy	72.0325%	75.4199%	88.1666%	88.9023%
	Loss	0.6589	0.6963	0.3227	0.3106
	TP	0.38	1.1933	0.8233	0.8
	TN	1.3799	1.06	1.82	1.8
	FP	0.62	0.42	0.1733	0.2
	FN	0.62	0.3184	0.1733	0.2
CNN+ LSTM	Accuracy	79.1458%	90.1328%	90.513%	90.9863%
	Loss	0.5148	0.3038	0.2712	0.2586
	TP	0.7333	0.89	0.86	0.8533
	TN	1.7333	1.8867	1.86	1.8566
	FP	0.2667	0.1066	0.1399	0.1499
	FN	0.2667	0.1066	0.14	0.1499
Bi-LSTM+ MSCNN	Accuracy	**96.6529%**	94.8645%	**97.2591%**	**97.2929%**
	Loss	0.1049	0.1375	0.0852	0.0976
	TP	0.9667	0.94	0.9667	0.9667
	TN	1.9667	1.9399	1.9667	1.9667
	FP	0.0333	0.06	0.0333	0.0333
	FN	0.0333	0.0599	0.0333	0.0333
Simple Bi-LSTM	Accuracy	94.7434%	**96.9804%**	93.1367%	93.3509%
	Loss	0.1401	0.1212	0.2233	0.1458
	TP	0.9367	0.96	0.92	0.9067
	TN	1,94	1.9533	1.9167	1.9067
	FP	0.0667	0.04	0.08	0.0933
	FN	0.0666	0.04	0.2899	0.0933

(FN). Evaluation was compiled from four non-overlapping chunks of data with different levels of applied noise, starting without any noise and ending with a 30 dB signal-to-noise ratio, the same as the last part of the training data, as shown in Fig. 5.12.

Table 5.2 shows the evaluation results, from which it can be seen that the best overall score was achieved by the Bi-LSTM + MSCNN model. This is most likely due to its ability to combine spatio-temporal features of backward and forward temporal looking capability of Bi-LSTM and spatial characteristics of MSCNN module. The worst performing model was the simple FC because it makes no assumptions about the spatio-temporal correlation of the input data, followed by the simple CNN whose performance was significantly better

but does not have the capability to learn temporal aspects of data. CNN + LSTM model showed initial poor performance, but gradually got better as it learned on noisy data. That is in correspondence with results from [95], which states that LSTM is less susceptible to perturbation in data if noise is present. This model could provide better results if LSTM layer is replaced by Bi-LSTM which would make it backward and forward looking capable. The simple Bi-LSTM model, that does not include the MSCNN part, has initially showed good accuracy until data with larger noise was introduced. Figure 5.13 shows the results of classification for all models, after having completed the whole learning process (60000 epochs), on the typical examples of lane change left and lane change right without added noise. According to Table 5.2, which contains five different architectures of neural networks, the final architecture recommendation would be: series of convolutional layers whose outputs are combined with input and feed into LSTM and then into series of fully connected layers followed by softmax. Best overall results were achieved with the network model composed of Bi-LSTM and MSCNN layers, achieving accuracy of up to 97.2%, due to its ability to learn spatio-temporal features of input data.

Consequently, the Bi-LSTM + MSCNN architecture, previously introduced in this section as part of the preliminary exploration of multiple network configurations for this task, was adopted as the starting point for the subsequent model design. To enhance its performance further, an expansion was implemented by introducing a direct connection from the output of each convolutional layer to the input of the *Bi-LSTM* layer, as shown in Fig. 5.14. An LSTM layer utilizes gated units with self-connected cells, allowing it to capture and propagate information over extended sequences while mitigating the vanishing gradient problem. Furthermore, the Bi-LSTM layer employs two LSTM networks, one processing input data in forward order and the other in reverse order, enabling it to capture context from both past and future inputs. In addition, an extra Bi-LSTM layer was added, followed by two fully connected layers. The input of the model is $n \times 10$, where n represents the number of data samples, while the second dimension consists of the vehicle's S,D coordinates followed by 8 lane lines. The initial three convolutional layers are responsible for extracting features from the input data. Each convolutional layer comprises 32 filters, with the first layer employing a kernel size of 5, and the subsequent two utilising a kernel size of 2. By merging the input and output of all convolutional layers, the data size increases to $n \times 10$.

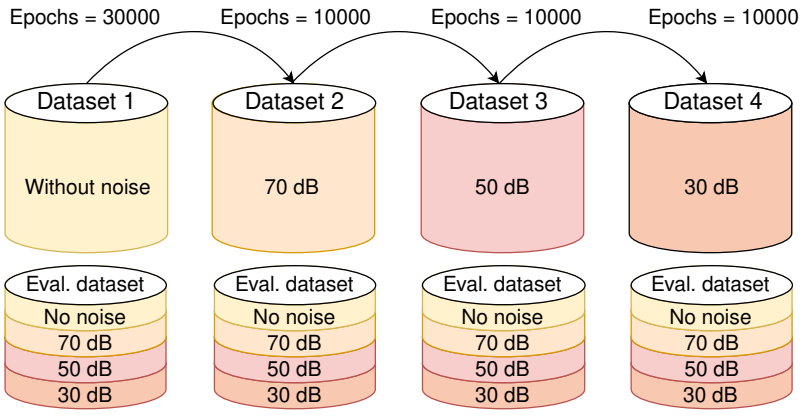
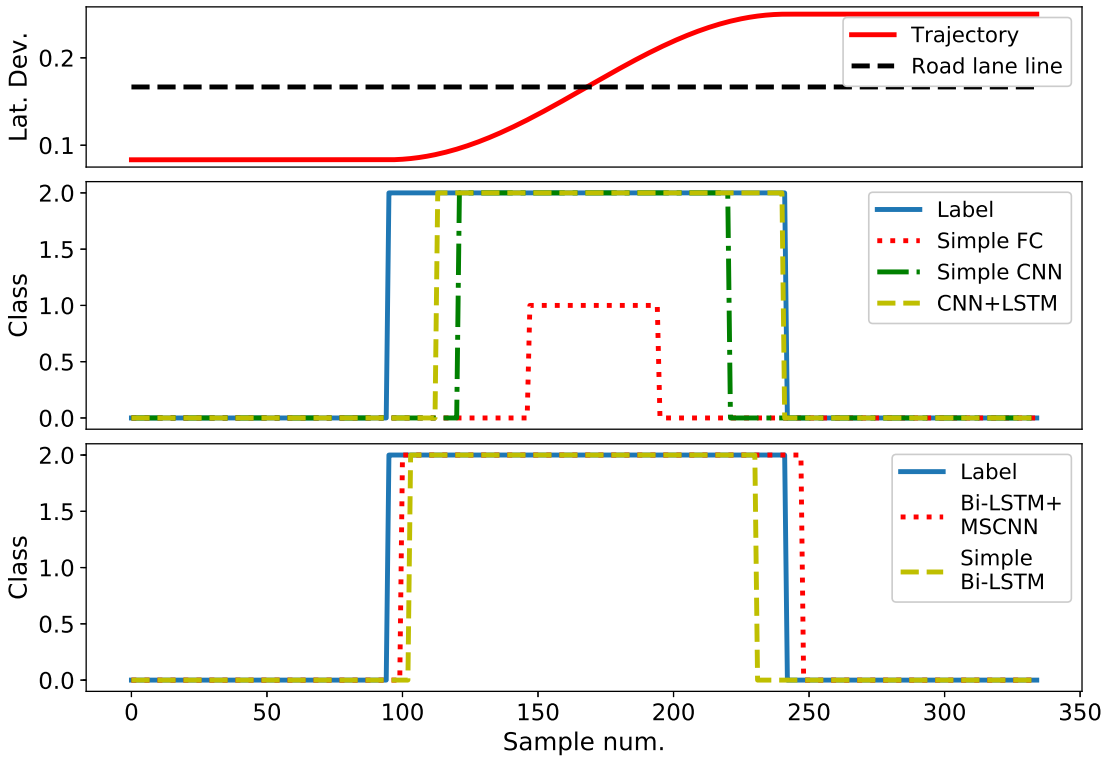
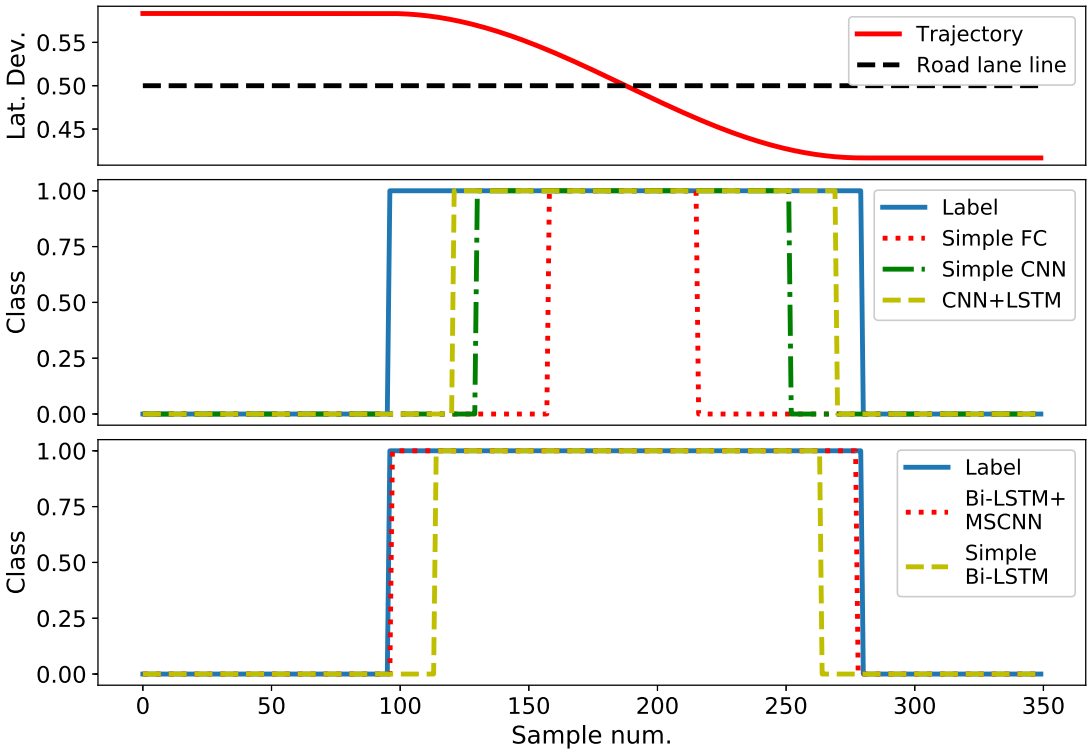


Figure 5.12: The dataset in the training process



(a) Lane Change Left



(b) Lane Change Right

Figure 5.13: Evaluation results

This amalgamation provides the subsequent layers with a more comprehensive view of both the input data and the features extracted by the convolutional layers. The two Bi-LSTM layers consist of 128 cells in the backward layer and 128 cells in the forward layer, resulting in an output of $n \times 256$. This layer gathers all the available data and prepares it for further processing in the subsequent layers of the model, which consist of two fully connected layers. The first of the two fully connected layers has 64 neurons, and the second layer has 32 neurons. The model's output is formed by a 3-neuron softmax layer, yielding an output size of $n \times 3$. In this context, n signifies the number of samples, and the second dimension gives the probability for each class.

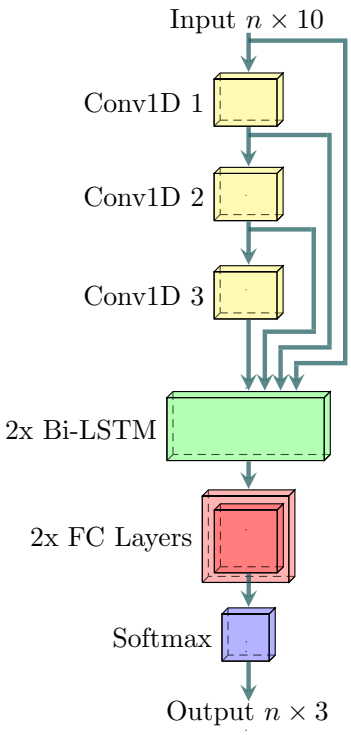


Figure 5.14: Model 5: Bi-LSTM + MSCNN

The *Dual Attention* model [87], the manoeuvre-classification stage of the LK-/LC-LSTM [88], and the sequence-to-sequence *TTS* model [89] were also implemented for comparison. All models had their input and output layers adjusted to match the dataset dimensions, ensuring a fair and consistent evaluation across all approaches. *Dual Attention* [87], except for input and output size adjustment, was not changed in any other respect. *TTS* [89] was modified so that the internal encoder-decoder LSTM produces a 3-class softmax output for variable-length inputs, replacing the original fixed-window binary classifier. Due to dataset limitations, only the manoeuvre-classification stage of the LK-/LC-LSTM framework [88] was used. In this form, it is a simple neural network with two feed-forward hidden layers, along with input and output layers, and is hereafter referred to as *Manoeuvre-Classification*.

Due to the limited availability of datasets containing annotated lane change actions, a novel dataset was curated and generated specifically for learning and evaluation. The dataset is the result of combining three datasets: HighD [67], NGSIM [68], and HLCsim [69]. Since

the HighD and NGSIM datasets do not contain labels for lane change actions, labels had to be added manually. The manual labelling of lane change actions involved placing markers in approximate areas corresponding to the start and end positions of the lane changes. While this method of data labelling may introduce minor imprecisions, it is acceptable given the non-deterministic nature of the conversion process. The HLCsim dataset already contains ground-truth lane-change labels generated by the simulation, which utilised lateral segments. The combination of the three datasets introduced a minor compromise in terms of label precision; however, it brings the advantage of using a blend of realistic and synthetic data for training and validation purposes. To broaden the dataset, it was enhanced through augmentation to improve data diversity and the noise robustness of the trained model [70]. Augmentation was applied in two steps, with the first step involving the mirroring of trajectories from left to right. This mirroring process produces the opposite lane change of the original, ensuring an equal number of lane change left and lane change right classes. In the second augmentation step, noise was applied at different signal-to-noise ratios: 70 dB, 50 dB, and 20 dB. The augmentation steps were exclusively applied to the training portion of the dataset, initially divided into an 80:20 training/validation ratio. The dataset's final size comprises 80448 trajectories for training and 2508 for validation; details on data composition and augmentations are shown in Table 4.1.

All aforementioned models were trained using the Adam optimizer [94] and the categorical cross-entropy loss function. The training and validation of the models were performed using the dataset shown in Table 4.1. Models were trained for 500 epochs, and those that achieved the best validation accuracy were selected for further analysis. Metrics for each model during training are shown in Figure 5.15, while the final training results are presented in Table 5.3. The metrics from the training process show that the *Bi-LSTM* model exhibits strong and consistent performance with minimal overfitting. It achieved the highest final training accuracy (99.847%), while its best validation accuracy ranked second (97.416%). The low standard deviation in validation accuracy (0.01080) indicates stable training and reliable generalisation. The Dual Attention model achieved the highest validation accuracy (97.586%) with a moderately high standard deviation (0.02660). Despite fluctuations, the model converged well, with a final validation accuracy of 97.334% and a training accuracy of 98.462%, indicating mild overfitting and overall good generalisation. The *TTS* model achieved high and stable performance, with the best validation accuracy of 96.722%. Training accuracy closely followed, ranking second with 96.804%, and the final accuracies were 97.739% (train) and 96.340% (validation). The validation accuracy standard deviation (0.01869) reflects a well-behaved training process with slight overfitting towards the end. The *Manoeuvre-Classification* model exhibits the lowest performance among the group but maintains consistent behaviour. Training and validation accuracies are nearly identical (84.836% and 85.277%, respectively), indicating no overfitting. The results confirm limited model capacity due to the shallow and simple model architecture. The validation portion of the dataset, described in Subsection 5.2.2 and summarised in Table 4.1, was used to evaluate all methods, including the polynomial-based approach (marked as Poly), the *Bi-LSTM* neural network, and the models proposed by other authors. The evaluation results are shown in Table 5.4, with novel metrics methods introduced in Section 4.3. The neural networks outperformed the polynomial-based approach, with the Dual Attention

Table 5.3: Validation and training performance metrics for each model at peak and final epochs

Model	Epoch	Val Acc	Val Loss	Train Acc	Train Loss
Bi-LSTM	112	0.97573	0.08536	0.98675	0.03165
	500	0.97416	0.24860	**0.99847**	**0.00368**
Dual_Att	361	**0.97586**	**0.07962**	0.98116	0.04703
	500	0.97334	0.09793	0.98462	0.03848
TTS	176	0.96722	0.10217	0.96804	0.08665
	500	0.96340	0.14479	0.97739	0.06131
Man_Class	480	0.85537	0.38463	0.84820	0.40418
	500	0.85277	0.38591	0.84836	0.40341

and *Bi-LSTM* models achieving the highest results. Without noise, the *Bi-LSTM* model achieved the best performance across all metrics. The polynomial-based method achieved better results than the *Manoeuvre-Classification* model across all metrics and outperformed the *TTS* model in all metrics except accuracy. By introducing noise corruption at a level of 70 dB, the Dual Attention model outperformed all others across all classification metrics, by a small margin closely followed by the *Bi-LSTM* model. The added noise did not change the ranking of the polynomial method, which still outperformed the *Manoeuvre-Classification* and *TTS* models in most metrics. Nevertheless, the polynomial method achieved better results under added noise across all metrics except Start/End. At the level of 50 dB, the *Bi-LSTM* model took the lead across all metrics, while the Dual Attention model ranked second. The polynomial method started to degrade under added noise, as all metrics decreased except for Centre Offset. It was no longer able to match the performance of the *TTS* and *Manoeuvre-Classification* models. By increasing the noise level to 20 dB, the *Bi-LSTM* model maintained the highest performance across all metrics. Its ability to handle high noise levels is attributed to the convolutional input layer, which provides a robust feature representation. The subsequent Bi-LSTM layers receive more informative inputs, enabling the model to classify data more accurately than the Dual Attention model, despite both architectures incorporating Bi-LSTM layers. The polynomial model, by contrast, performed significantly worse across all metrics except for Centre Offset. This indicates that it remains effective at estimating the centre position of the lane change but performs poorly on other evaluation criteria. This behaviour results from the model's use of polynomial representations for both vehicle trajectories and lane lines, combined with a criterion function that emphasises curve fitting rather than the longitudinal extent of the lane change. As a consequence, under increased noise, the polynomial method tends to narrow the estimated start and end points of the lane change, bringing them closer into alignment with the lane line crossing point. Overall, the *Bi-LSTM* model performed best, followed by the Dual Attention model. The *TTS* and *Manoeuvre-Classification* models took third and fourth place respectively in terms of accuracy, while the polynomial method ranked last. However, when considering other metrics, the polynomial method achieved better results than both the *TTS* and *Manoeuvre-Classification* models. Additionally, a visual summary of the results for

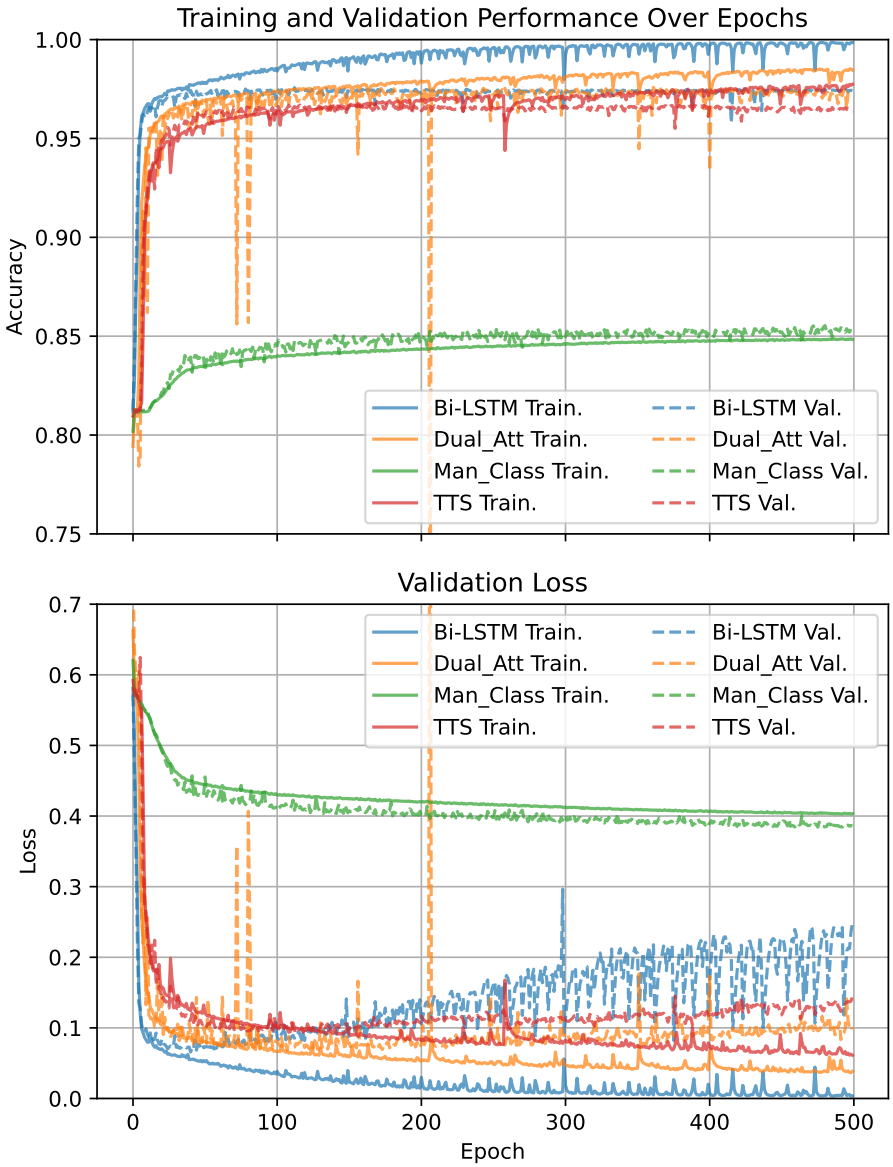


Figure 5.15: Validation accuracy and loss during learning process

each method and each data subset (HighD, HLCsim, NGSIM) can be seen in Figs. 5.16–5.18, where each segment is represented by a single entry. Each subplot in Figs. 5.16–5.18 represents a different level of noise, ranging from no added noise up to 20 dB. The best overall results were achieved by the *Bi-LSTM* model presented in this work, closely followed by the Dual Attention model [87]. Two other models, *TTS* [89] and *Manoeuvre-Classification* [88], performed worse and were outperformed by the Polynomial method according to the metrics introduced in this work. The evaluation results suggest that, in broader applications, the neural network models consistently attain superior performance across all parameters compared to the Polynomial method. Notably, the performance disparities between the neural network models and the Polynomial method significantly intensify as the input noise level increases. However, these results were obtained using validation data very similar to the training data, which may not reflect real-world conditions and gives an unfair advantage to the neural network models. The performance of neural networks depends

Table 5.4: Classification assessment on combined dataset

Noise level	Method	Accuracy	Start/ End	Centre offset	Estimate Usability
w/o	Poly	0.88250	0.89881	0.90619	0.87449
	Bi-LSTM	**0.97402**	**0.96759**	**0.97729**	**0.97965**
	Dual_Att	0.97323	0.96572	0.97623	0.97881
	TTS	0.95691	0.88843	0.90112	0.88919
	Man_Class	0.85826	0.83517	0.88754	0.86644
70 dB	Poly	0.88476	0.89741	0.91668	0.89104
	Bi-LSTM	0.97395	0.96759	0.97729	0.97964
	Dual_Att	**0.97321**	**0.96572**	**0.97622**	**0.97880**
	TTS	0.95639	0.88687	0.90067	0.88291
	Man_Class.	0.85819	0.83388	0.88707	0.86347
50 dB	Poly	0.81050	0.85581	0.92432	0.77310
	Bi-LSTM	**0.97334**	**0.96826**	**0.97816**	**0.98073**
	Dual_Att	0.97245	0.96518	0.97591	0.97853
	TTS	0.95081	0.82029	0.84306	0.78808
	Man_Class	0.85666	0.79914	0.88143	0.73651
20 dB	Poly	0.15507	0.54076	0.64969	0.07850
	Bi-LSTM	**0.94385**	**0.80274**	**0.83076**	**0.74348**
	Dual_Att	0.93992	0.81405	0.83710	0.78942
	TTS	0.92416	0.81422	0.85347	0.78826
	Man_Class	0.83982	0.55767	0.65792	0.26185
avg.	Poly	0.68321	0.79820	0.84922	0.65428
	Bi-LSTM	**0.96531**	**0.92937**	**0.94246**	**0.93236**
	Dual_Att	0.96076	0.92771	0.94546	0.93110
	TTS	0.92598	0.78832	0.82569	0.70551
	Man_Class	0.83953	0.73820	0.82469	0.63005

on the characteristics of the input data and may require additional training to adapt to different noise levels. The polynomial method is parameterisable and more likely to provide continuous and centred labels, making it better suited to low-noise data. The remainder of this thesis focuses on comparing the performance of the Bi-LSTM and polynomial methods in complete logic scenario reconstruction. The suggested metrics provide additional insights into classification quality. One particularly informative metric was *Centre Offset*, as it shows the ability of the polynomial method to detect the centre of the lane change. However, a notable disadvantage is its reliance on labelled data, which limits its application to datasets that include labels.

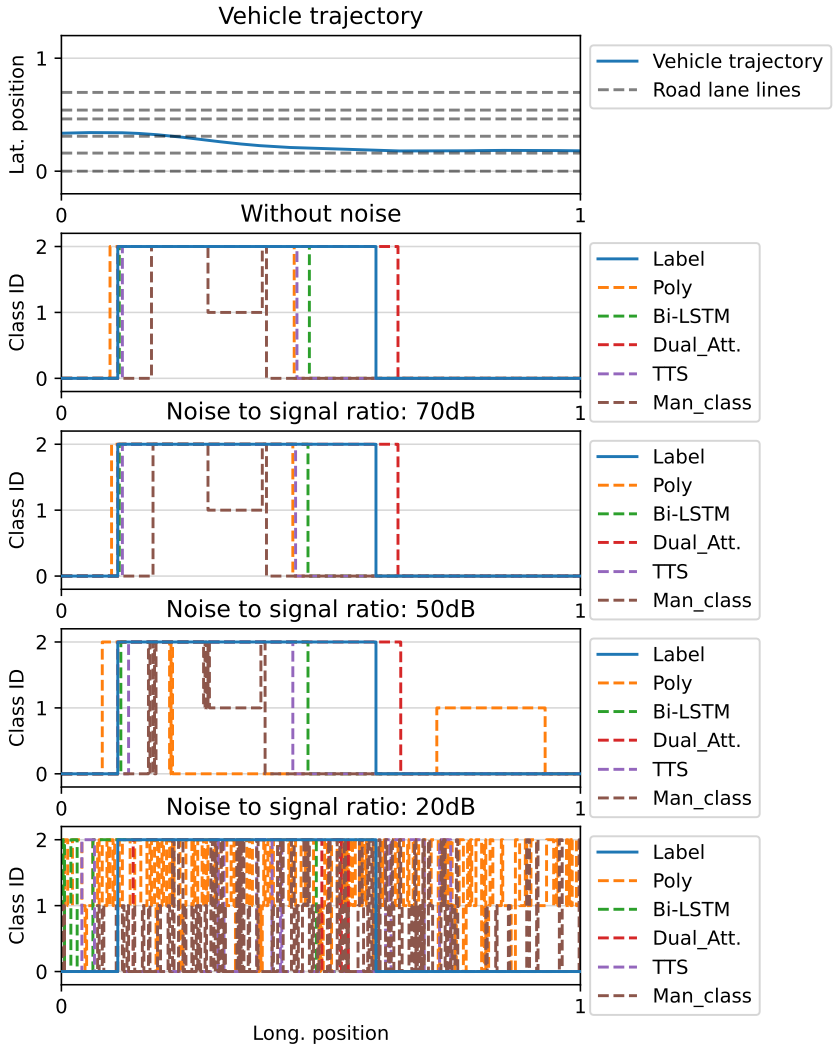


Figure 5.16: Evaluation example HighD

5.3 Classification of Longitudinal Vehicle Actions

This section provides a detailed description of how the longitudinal component of the vehicle action classification problem, introduced in Section 1.1, has been implemented in this thesis. The implementation builds directly on the greedy knot reduction method described in Subsection 2.4.2, applying it to approximate longitudinal vehicle actions with quadratic splines.

5.3.1 *Greedy Knot Constant-Acceleration Segments*

Longitudinal vehicle actions can be classified and approximated using piecewise constant-acceleration segments. This approach produces a compact and interpretable trajectory description, where phases of acceleration, braking, and constant velocity are clearly distinguished. This is implemented by an algorithm that starts by fitting an initial quadratic spline through all data points. During each iteration, it evaluates the effect of removing one interior knot on the approximation error, defined in terms of the supremum norm, and compares this error increase to the given cost value. If the error increase is below the

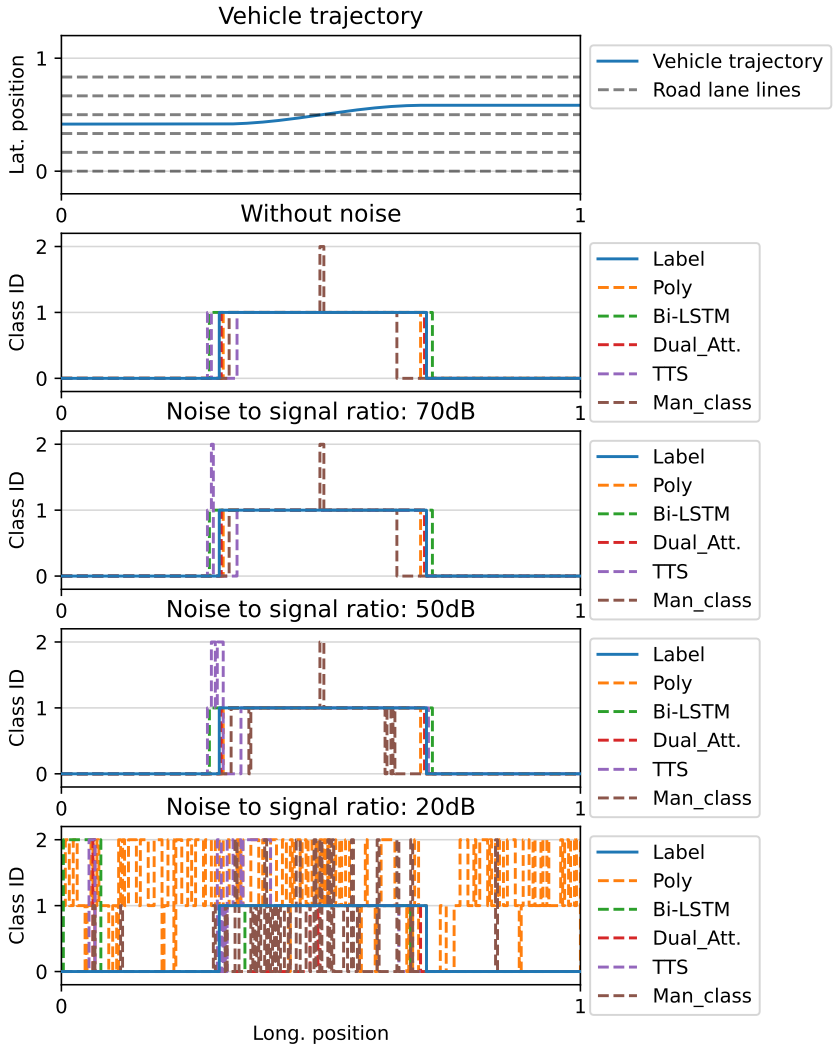


Figure 5.17: Evaluation example HLCsim

cost value, the knot is removed. At each iteration, the knot whose removal produces the smallest error increase is selected, which makes the algorithm greedy. This iterative process continues until the minimal requested number N of segments is reached or no further removals satisfy the error criterion. The error is measured by the squared residual norm defined in Eq. 2.6. For a candidate knot i, the criterion function is expressed as

$$\Delta E(i) = \sum_{k=1}^{N} \left(y_k - S_{-i}(x_k)\right)^2 - \sum_{k=1}^{N} \left(y_k - S(x_k)\right)^2, \tag{5.10}$$

where $S(x)$ is the spline approximation with the current knot sequence (see Eq. 2.5) and $S_{-i}(x)$ is the spline obtained after removing knot i. A knot is removed if

$$\Delta E(i) < w, \tag{5.11}$$

where w denotes the cost associated with adding or retaining a knot. At each iteration, the algorithm selects the knot

$$i^* = \arg\min_i \Delta E(i), \tag{5.12}$$

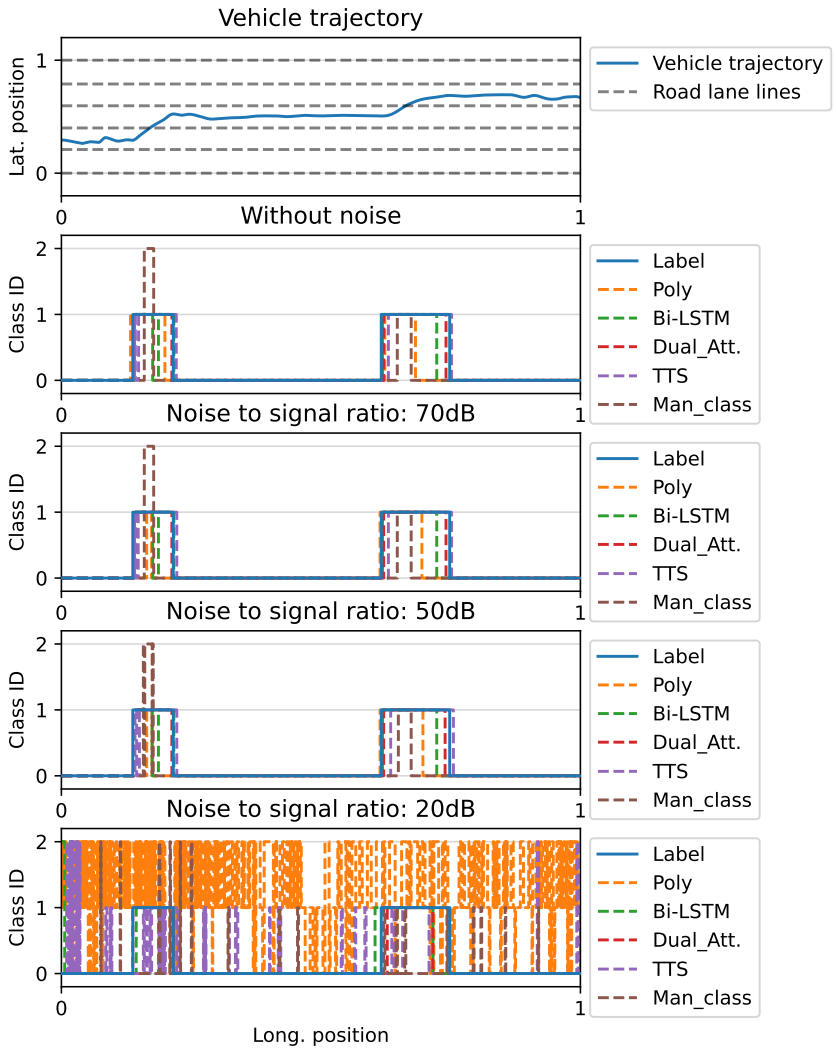


Figure 5.18: Evaluation example NGSIM

in accordance with the greedy update rule in Eq. 2.7–2.8. In this way, the algorithm provides a method to compress a trajectory into a reduced spline representation while balancing fidelity and simplicity. This approach offers several advantages for longitudinal action classification. By representing the trajectory with a minimal set of constant-acceleration segments, it becomes possible to identify distinct phases of driving behaviour such as acceleration, deceleration, and cruising. Each segment corresponds to a period where the vehicle maintains approximately constant acceleration, providing a compact yet interpretable description of the longitudinal motion. This not only reduces the dimensionality of the data but also aligns directly with the motion primitives typically used in vehicle dynamics and scenario-based simulation. Moreover, the greedy knot reduction method ensures that the segmentation is data-driven, adapting to the complexity of the observed trajectory. Trajectories with smooth behaviour require fewer segments, while trajectories with frequent changes in acceleration retain more knots to preserve accuracy. The supremum norm error criterion further guarantees that the approximation respects the most critical deviations from the original signal, which is essential for maintaining fidelity in safety-critical applications. Formally, each segment can be classified based on its constant acceleration value $\ddot{x}$ as follows:

- $\ddot{x} > 0 \Rightarrow$ Acceleration :
- $\ddot{x} < 0 \Rightarrow$ Braking :
- $\ddot{x} \approx 0 \Rightarrow$ Constant velocity :

The resulting segmentation serves as the foundation for classifying longitudinal actions in the context of logic scenario generation. Each constant-acceleration segment can be directly labelled and transformed into a standardized description of the vehicle's motion. This classification not only simplifies the interpretation of recorded driving data but also facilitates its transformation into reusable logic scenarios for validation and testing of automated driving systems.

5.4 LOGIC SEGMENTS FORMATION

A logic scenario is a conditionally driven sequence of logic segments that describes vehicle actions, as outlined in Chapter 2. Each logic segment consists of parameters that describe the vehicle action in that segment and the corresponding segment-ending conditions. To support most simulator software, segment-ending conditions are restricted to time-based durations. Lateral vehicle actions are defined as lane change left, lane change right, or lane keeping, while longitudinal actions are defined using constant-acceleration segments. This section describes how to form a logic scenario, along with its corresponding segments and parameters, from a classified vehicle trajectory.

5.4.1 *Initial Conditions*

A logic scenario starts with the initial conditions, including longitudinal position, initial vehicle velocity, initial lane, lateral offset from the lane centreline, vehicle orientation, and similar quantities. Initial lane, lateral offset from the centreline, and the vehicle's initial longitudinal position are obtained from the first few samples of the vehicle trajectory and the road lane lines. The initial lane is determined by comparing the road lane lines to the left and right of the vehicle position in the curvilinear coordinate system, while the lateral offset is computed by comparing the vehicle position with the lane centreline. Longitudinal vehicle position is usually taken as the first S coordinate in the vehicle trajectory. Initial vehicle velocity is determined from the first classified vehicle action. By comparing the estimated acceleration for the given action, its duration, and the initial and final positions, a standard constant-acceleration position formula is used:

$$x_f = x_0 + v_0\, t + \tfrac{1}{2} a t^2. \tag{5.13}$$

Subtracting the initial position x_0 and $\frac{1}{2}at^2$ from the right side, and dividing by the segment duration, the initial velocity v_0 can be expressed as:

$$v_0 = \frac{x_f - x_0}{t} - \tfrac{1}{2} a t. \tag{5.14}$$

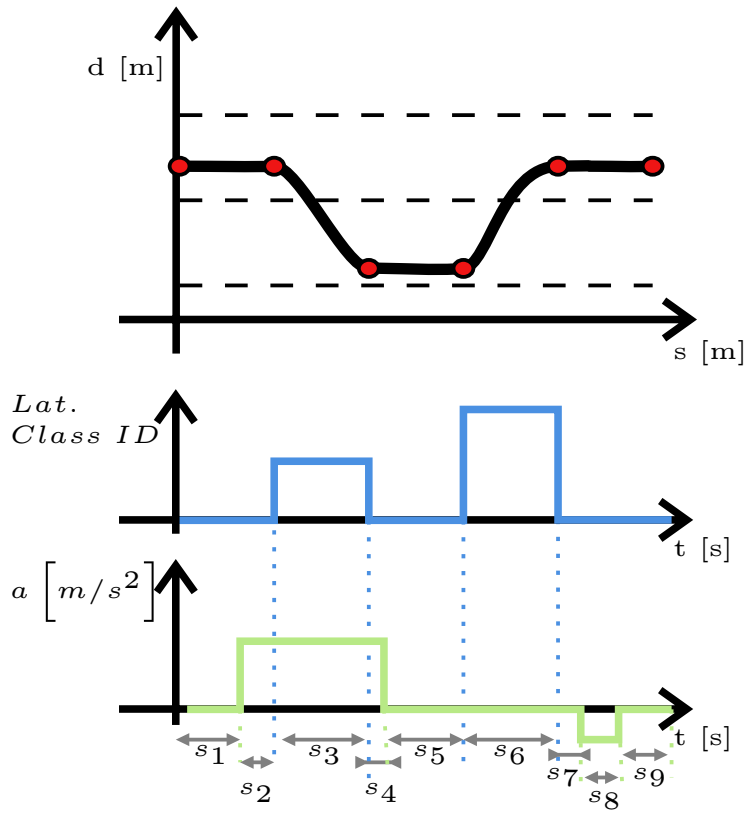


Figure 5.19: Forming a logic scenario

5.4.2 *Segment Forming*

Segment forming starts with an analysis of the results from the longitudinal and lateral classifications. Actions from both classifications are cross-paired and merged into segments with distinct lateral and longitudinal actions. Every classified vehicle action from either classification results in the formation of a new segment. This procedure can be described as an XOR boolean operation between the longitudinal and lateral classification results. An example is shown in Fig. 5.19, where the defined segments are marked with s_1 to s_9.

These segments represent distinctive action intervals that sequentially describe all vehicle actions in the input trajectory. Each segment duration is taken from the time vector, and representative longitudinal acceleration and velocity values are sampled near the start of each segment. The lateral and longitudinal parameters of each segment are obtained from the respective classifications. A logic scenario object is then initialized with the appropriate initial conditions, including longitudinal offset, initial speed, and initial lane.

5.5 Evaluation at the Scenario Level

This section applies both the neural-network and polynomial lateral classification methods, combined with greedy knot-based longitudinal classification, to evaluate the performances of each approach in driving scenarios. Two types of evaluation experiments were conducted. The first uses synthetic data, as described in Subsection 5.5.1, while the second uses real-world driving trajectories from the HighD dataset, presented in Subsection 5.5.2. For both

methods, the trajectories were modified with different levels of distortion and analysed with and without the novel filtering method introduced in Chapter 3.

5.5.1 *Reference Logical Scenarios*

Testing the accuracy of the classification method described earlier in this chapter requires both ground-truth logical scenarios and trajectory data. This is achieved by evaluating the complete conversion process on the reference dataset [71], which is described in Subsection 4.2.5. Figure 5.20 illustrates this evaluation process in more detail. The dataset starts with three basic logical scenarios, SimpleLC, DoubleLC, and UshapeLC, shown in Figure 4.3. These three scenarios are used for testing. Each scenario has multiple test cases (TCs) with different segment parameters, including longitudinal acceleration, lateral velocity, and segment duration, as listed in Table 4.2. In addition, for each test case, multiple distortion configurations (DCs), listed in Table 4.3, were applied to evaluate the robustness of the classification method and to determine whether applying simulation-based filtering during preprocessing improves classification performance. Overall, these logical scenarios comprise 96 publicly available evaluation experiments under the name *LC Scenarios for Lane Change Classification* [71]. Each experiment was run with and without simulation-based filtering, and both polynomial and neural-network classification methods were applied, yielding a total of 384 evaluation runs.

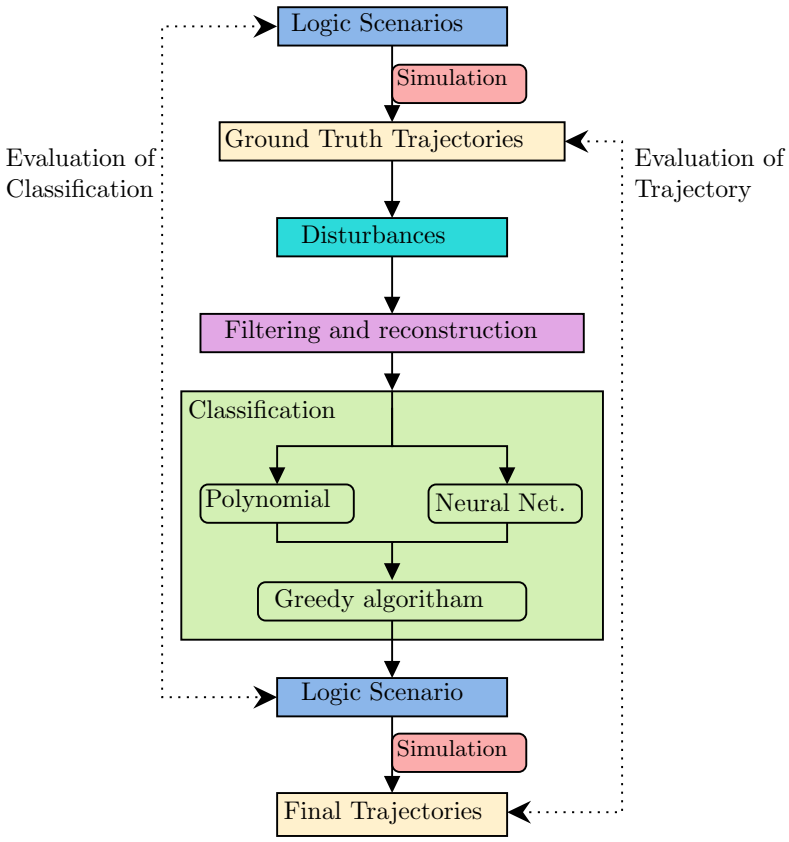


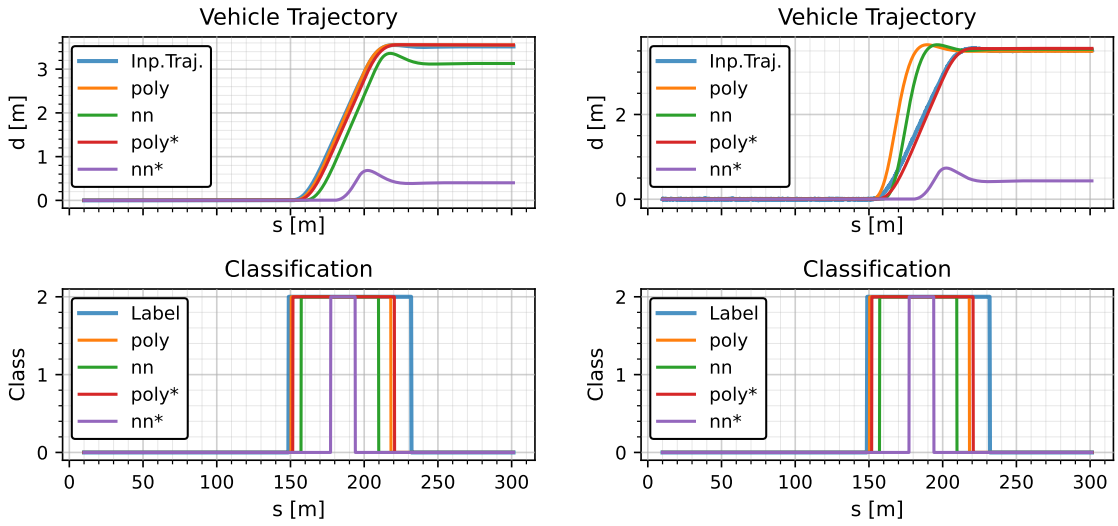
Figure 5.20: Logic scenarios evaluation process diagram

Results of the evaluation are presented in Tables 5.4 and in Figures 5.16-5.17, with metrics for both the classification and trajectory components reported separately for each test case

Table 5.5: Evaluation Results SingleLC across all test cases

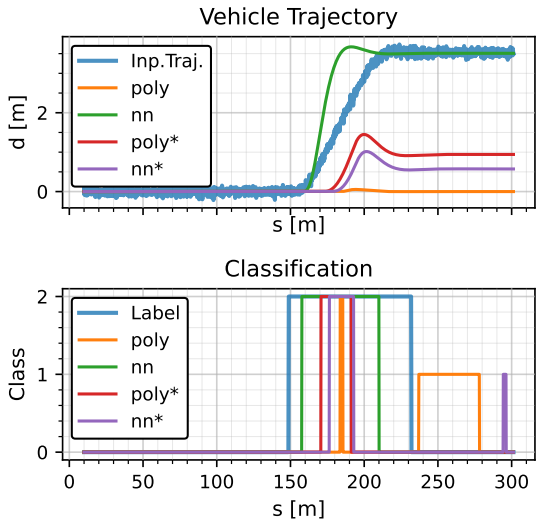
SingleLC		All test case							
Distr. config:		DC1	DC2	DC3	DC4	DC5	DC6	DC7	DC8
Classification	Label Accuracy	poly	0.94669	**0.94721**	0.57983	0.36319	0.72011	**0.94764**	0.36448	0.71725
		nn	0.89480	0.89480	**0.89485**	**0.89480**	0.85577	0.89623	0.85482	0.90337
		poly*	**0.94721**	0.94669	0.78342	0.81483	**0.94721**	0.94683	**0.93783**	**0.93450**
		nn*	0.77056	0.77094	0.76333	0.77456	0.77047	0.77113	0.77085	0.76942
	Start Ends	poly	0.94669	**0.94717**	0.71585	0.72394	0.71966	**0.94764**	0.73536	0.71775
		nn	0.89481	0.89481	**0.89481**	**0.89481**	0.85388	0.89624	0.85055	0.89814
		poly*	**0.94717**	0.94669	0.78344	0.81485	**0.94764**	0.94717	**0.93765**	**0.93432**
		nn*	0.77059	0.77106	0.77011	0.78867	0.77059	0.77154	0.77106	0.76963
	Centers Offset	poly	**0.97953**	**0.97977**	**0.98358**	**0.98287**	0.97596	**0.97953**	0.96668	0.97501
		nn	0.97692	0.97692	0.97834	0.97692	**0.98263**	0.97763	**0.98144**	0.97382
		poly*	0.97882	0.97953	0.95883	0.96264	0.97906	0.97882	0.97358	**0.97763**
		nn*	0.97715	0.97692	0.97120	0.95954	0.97715	0.97525	0.97216	0.97573
	Label Usage	poly	**1.00**	**1.00**	0.00971	0.02522	**1.00**	**1.00**	0.05207	**1.00**
		nn	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
		poly*	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
		nn*	**1.00**	**1.00**	0.89313	0.83871	**1.00**	**1.00**	**1.00**	**1.00**
	Segment Overhead[%]	poly	**11.11**	**11.11**	38.46154	66.66667	**11.11**	**11.11**	46.66667	**11.11**
		nn	**11.11**	**11.11**	**11.11**	**11.11**	**11.11**	**11.11**	**11.11**	**11.11**
		poly*	27.27273	27.27273	27.27273	27.27273	27.27273	**11.11**	27.27273	27.27273
		nn*	27.27273	27.27273	38.46154	38.46154	27.27273	**11.11**	27.27273	27.27273
Trajectory	mse	poly	**0.00350**	0.29324	4.60989	35.94059	4.62497	0.44844	8.10103	4.62582
		nn	0.08514	**0.13920**	**0.22841**	**0.23833**	**0.60631**	0.23987	**0.10369**	**0.30451**
		poly*	4.01697	4.04805	10.02989	13.21887	3.73088	**0.01739**	3.58506	3.87000
		nn*	7.83092	7.72732	10.99953	14.53723	7.53928	3.54849	8.02771	7.33078
	std	poly	**0.06603**	0.56332	1.67944	3.72674	1.70079	0.70056	2.05829	1.69872
		nn	0.25073	**0.40001**	**0.51584**	**0.52561**	**0.60066**	0.52756	**0.36048**	**0.59148**
		poly*	1.21854	1.23423	2.78002	2.42350	1.18371	**0.14727**	1.31015	1.32185
		nn*	2.65136	2.63300	2.97432	2.93208	2.61572	1.58204	2.76337	2.61709

(TC). The polynomial method is denoted by *poly* and the neural-network model by *nn*, while an asterisk () marks entries for which simulation-based filtering was applied during preprocessing.

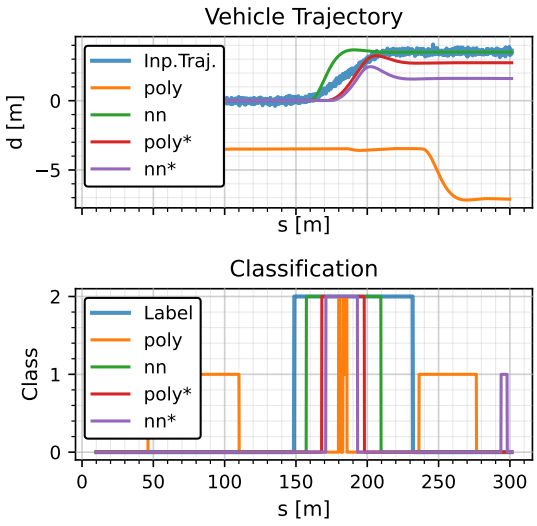


(a) DC1

(b) DC2

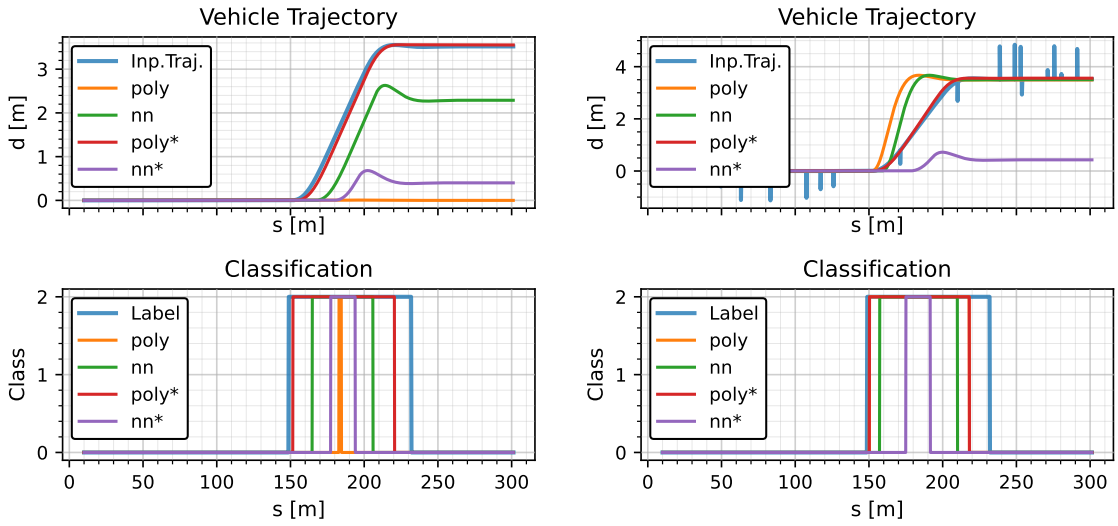


(c) DC3



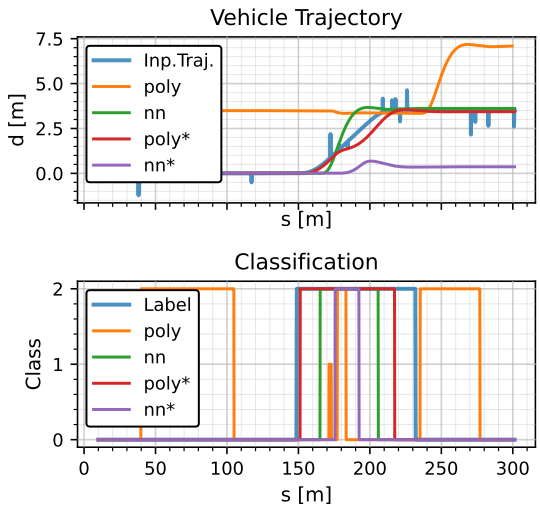
(d) DC4

Figure 5.21: Trajectory classification results for SingleLCTC1 (DC1-DC4).

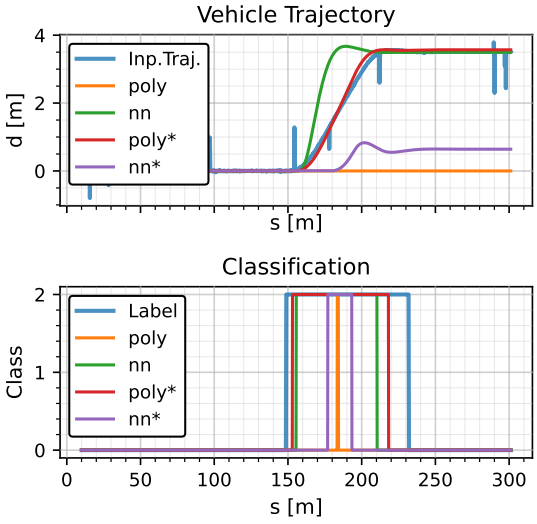


(a) DC5

(b) DC6

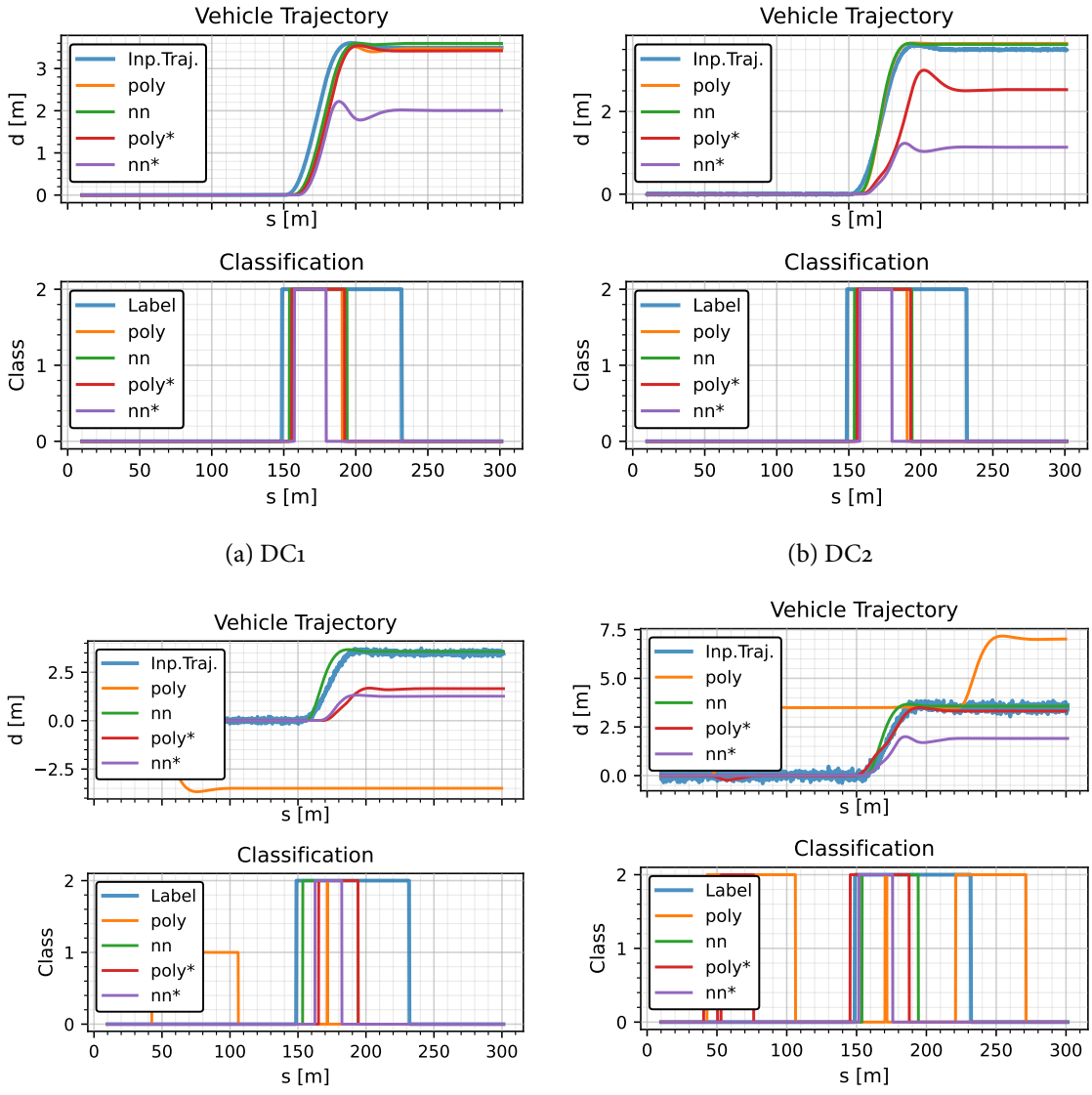


(c) DC7



(d) DC8

Figure 5.22: Trajectory classification results for SingleLCTC1 (DC5-DC8).



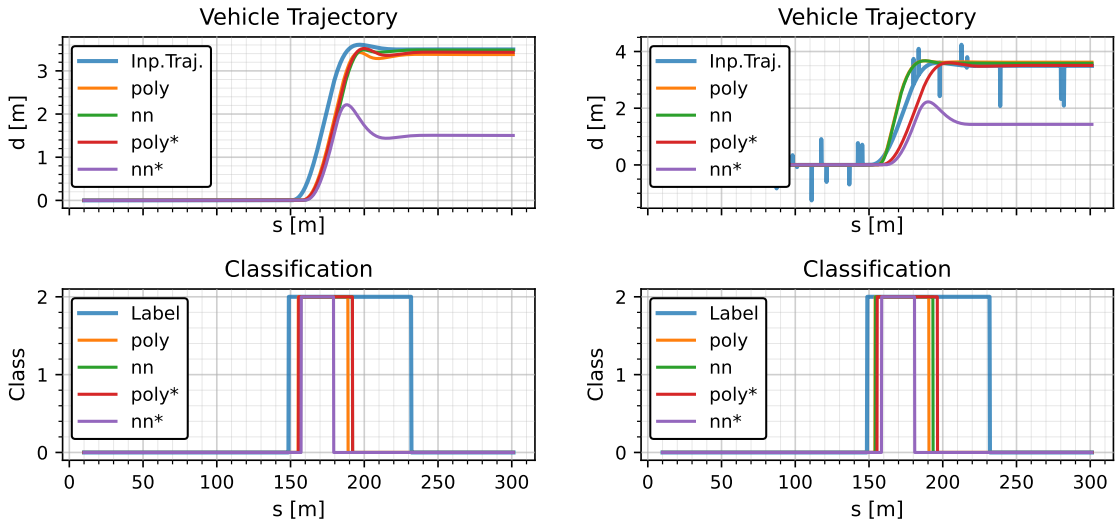
(a) DC1

(b) DC2

(c) DC3

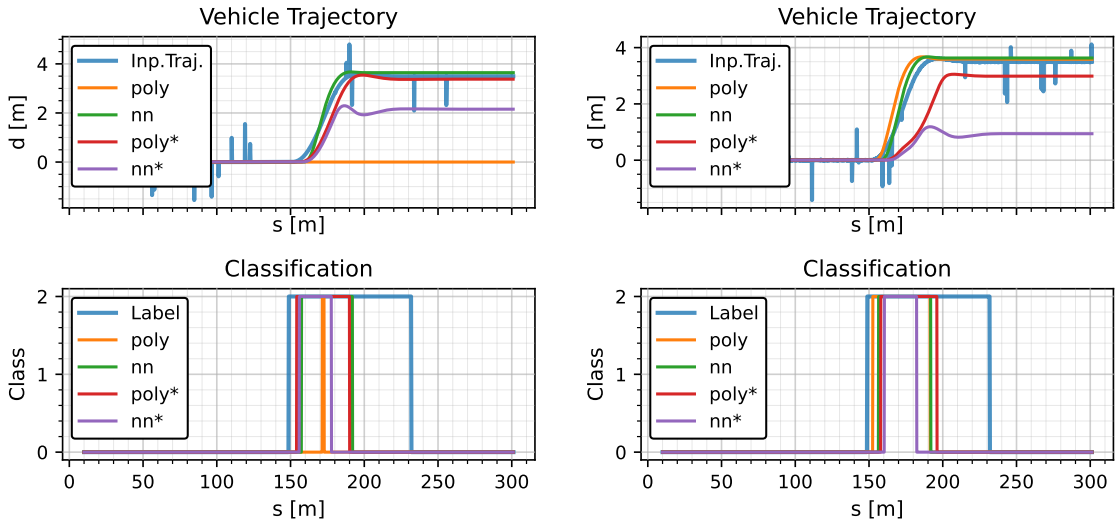
(d) DC4

Figure 5.23: Trajectory classification results for SingleLCTC2 (DC1-DC4).



(a) DC5

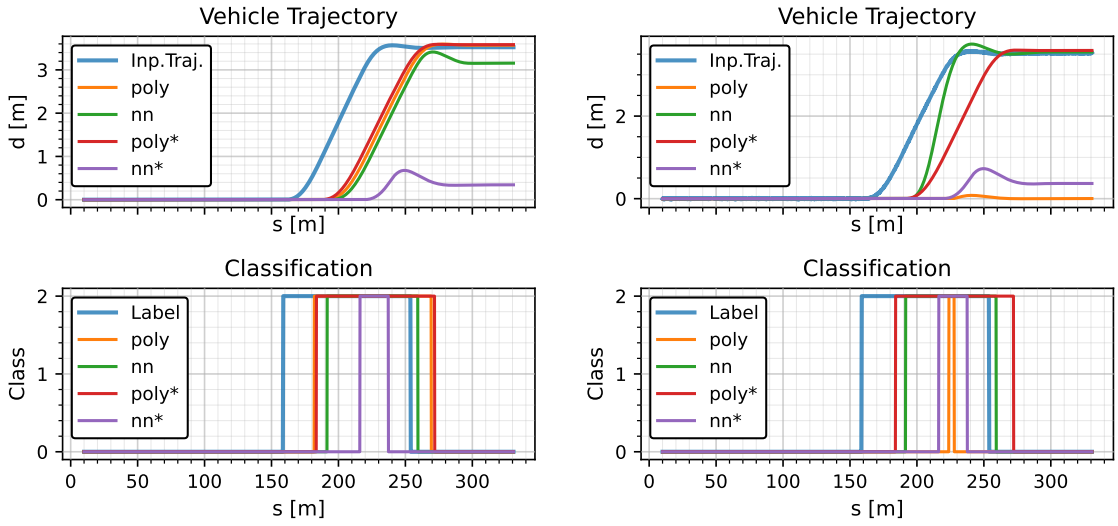
(b) DC6



(c) DC7

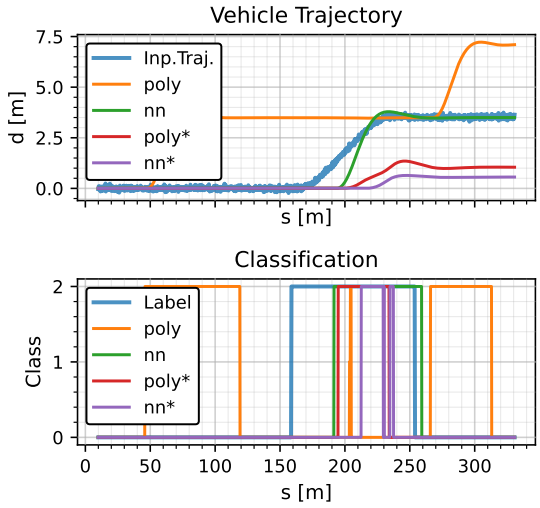
(d) DC8

Figure 5.24: Trajectory classification results for SingleLCTC2 (DC5-DC8).

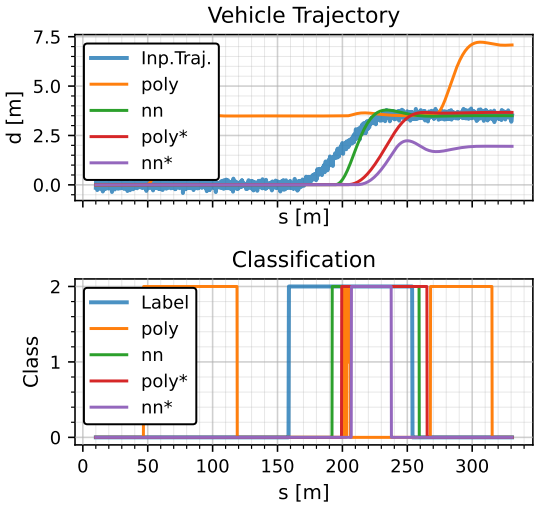


(a) DC1

(b) DC2

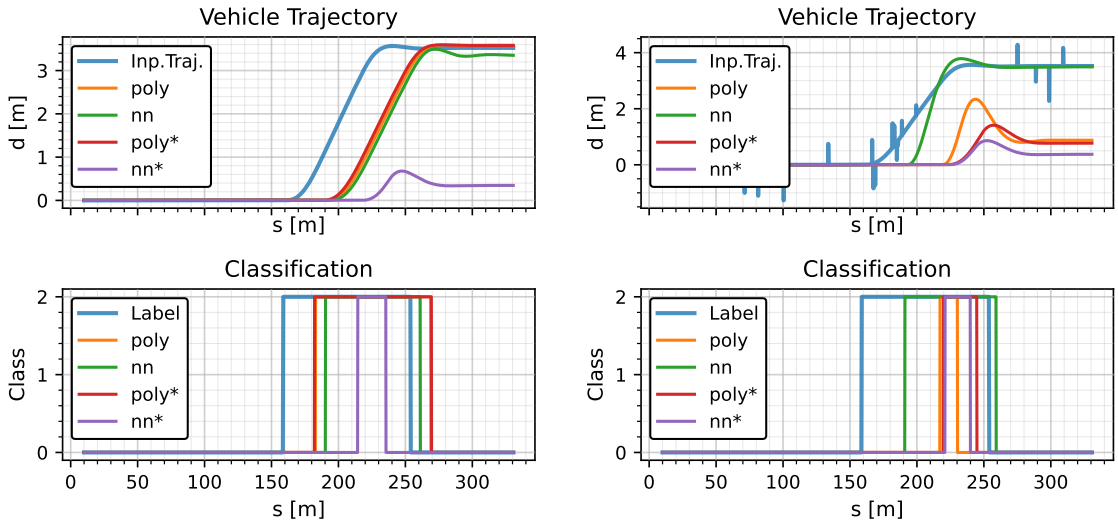


(c) DC3



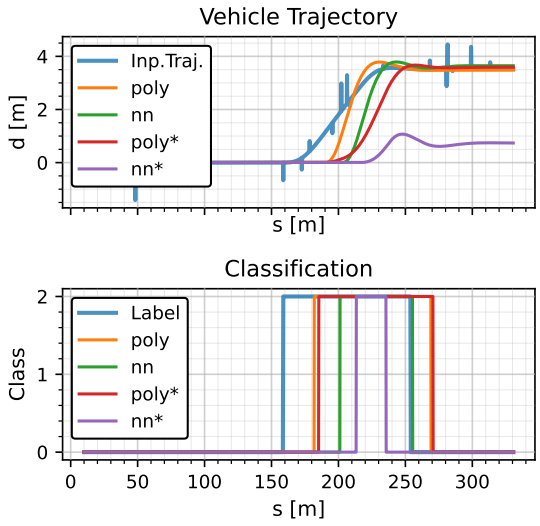
(d) DC4

Figure 5.25: Trajectory classification results for SingleLCTC3 (DC1-DC4).

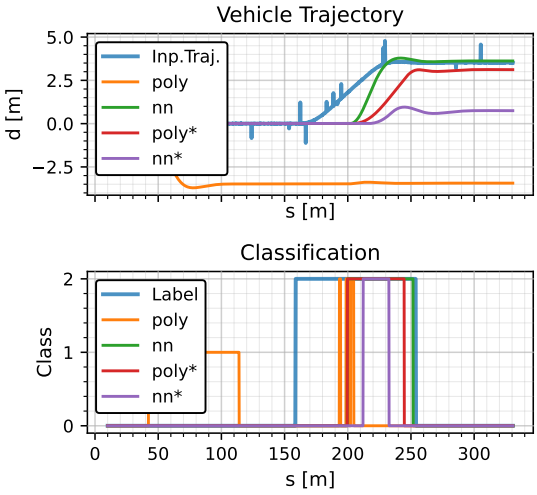


(a) DC5

(b) DC6

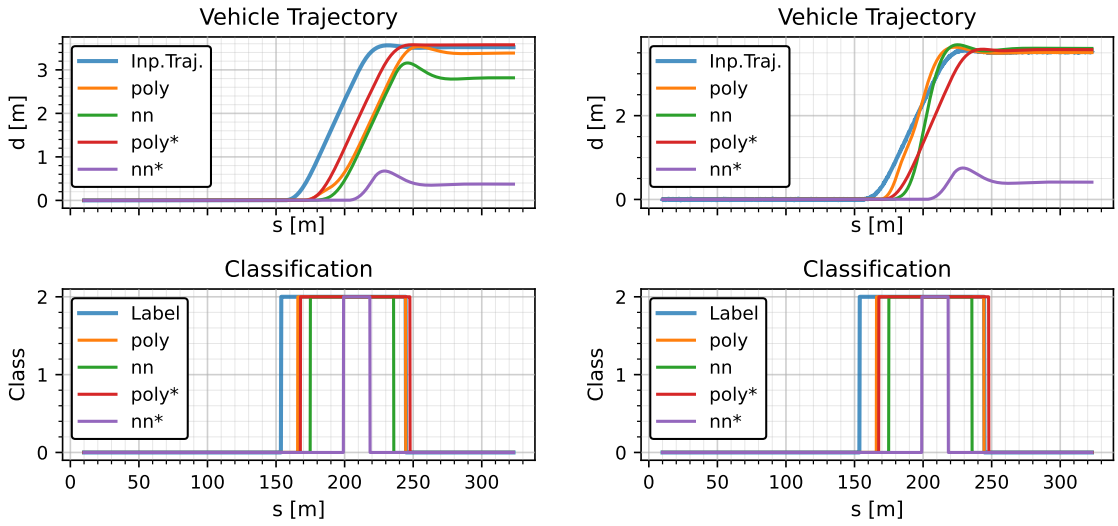


(c) DC7



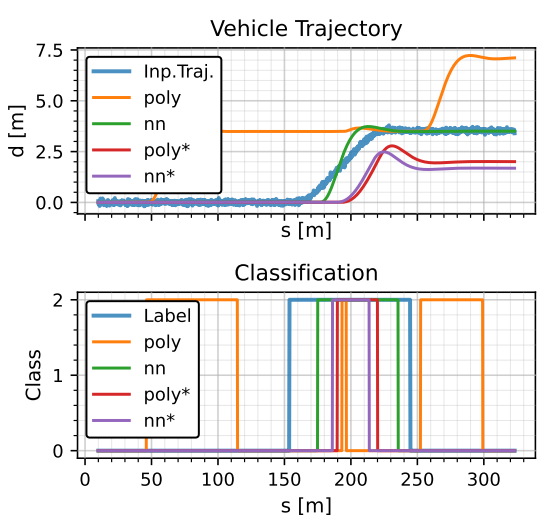
(d) DC8

Figure 5.26: Trajectory classification results for SingleLCTC3 (DC5-DC8).

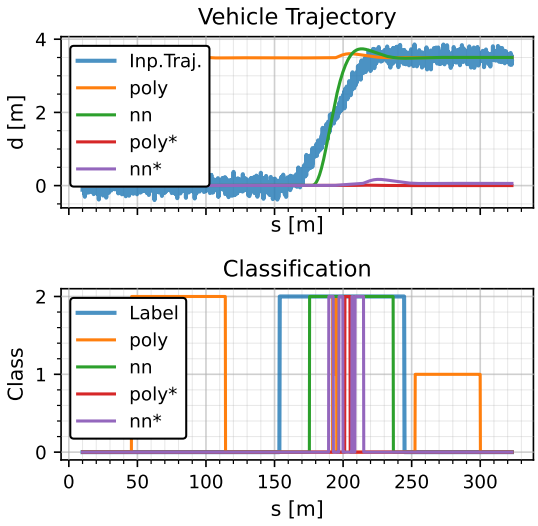


(a) DC1

(b) DC2

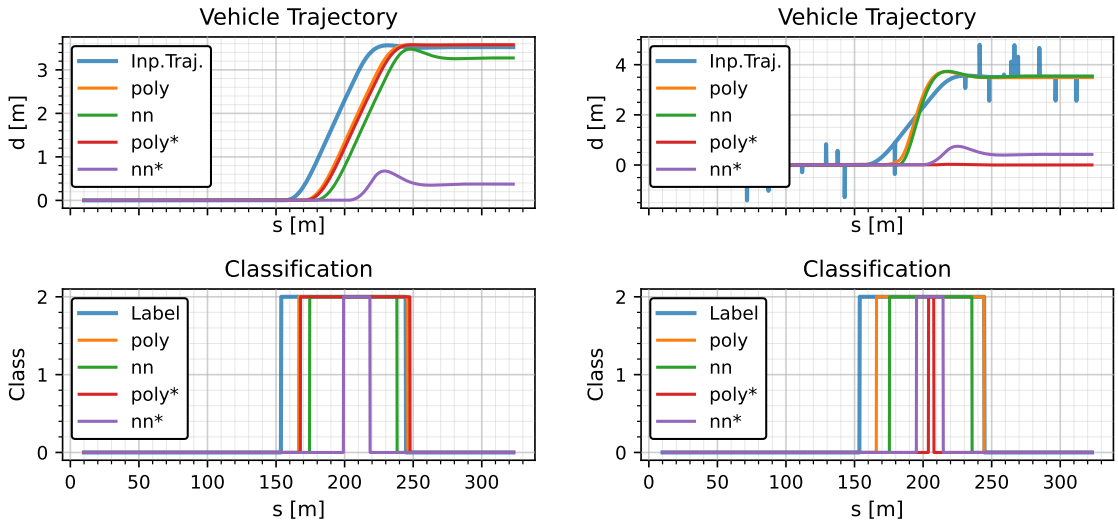


(c) DC3



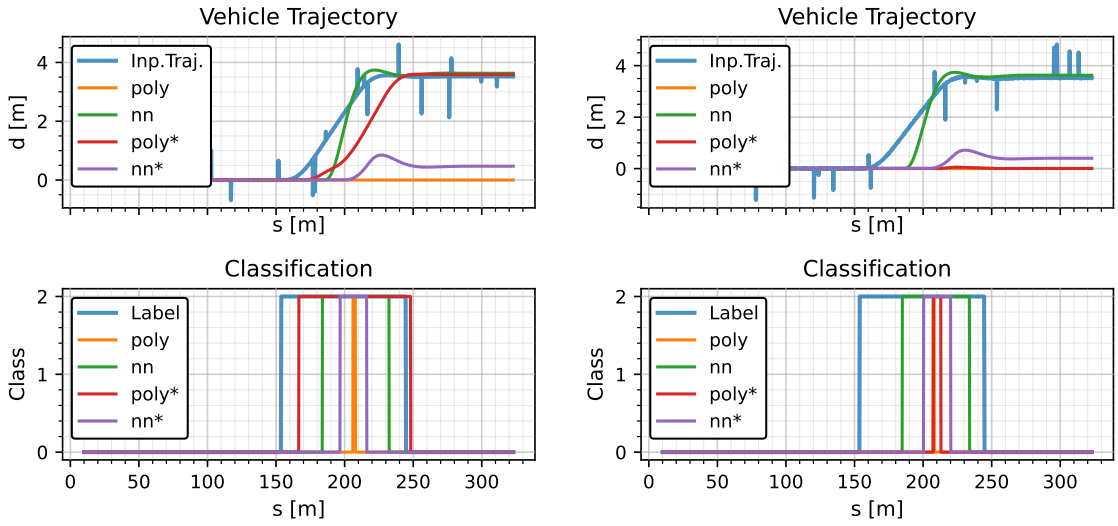
(d) DC4

Figure 5.27: Trajectory classification results for SingleLCTC4 (DC1-DC4).



(a) DC5

(b) DC6



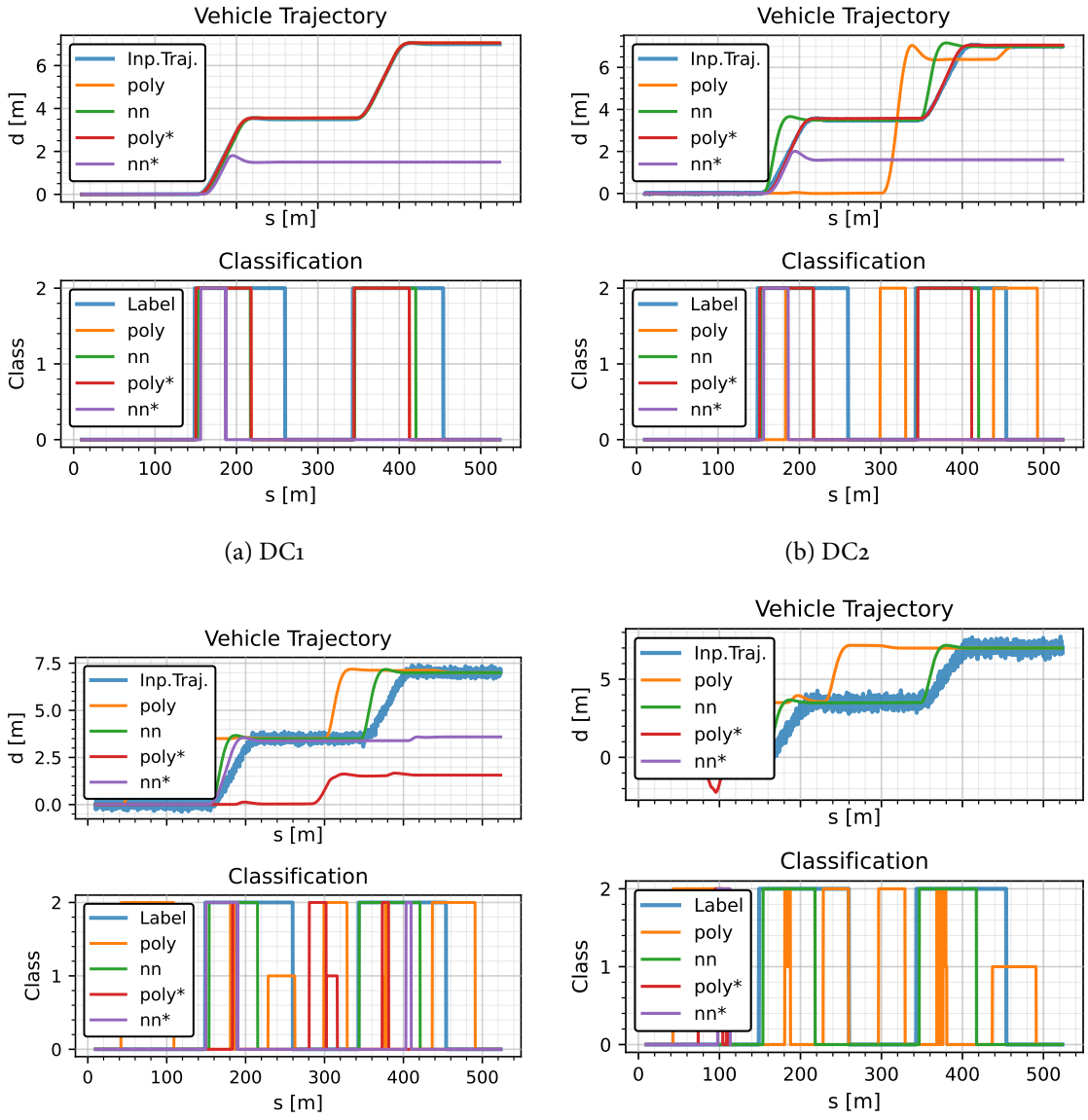
(c) DC7

(d) DC8

Figure 5.28: Trajectory classification results for SingleLCTC4 (DC5-DC8).

Table 5.6: Evaluation Results DoubleLC across all test cases

DoubleLC			All test case							
Distr. config:			DC1	DC2	DC3	DC4	DC5	DC6	DC7	DC8
Classification	Label Accuracy	poly	0.83167	0.46144	0.33191	0.37590	0.83332	0.53737	0.51694	0.40141
		nn	**0.83956**	**0.83953**	**0.83842**	**0.83113**	**0.84628**	**0.83688**	**0.83604**	**0.84469**
		poly*	0.83194	0.82551	0.52753	0.50743	0.83194	0.56763	0.56763	0.56763
		nn*	0.62745	0.62651	0.65812	0.45563	0.62751	0.56763	0.56763	0.56763
	Start Ends	poly	0.82850	0.56286	0.56095	0.56413	0.82427	0.56483	0.56190	0.56135
		nn	**0.83929**	**0.83929**	**0.83953**	**0.82959**	**0.84058**	**0.84614**	**0.83308**	**0.84003**
		poly*	0.82874	0.82206	0.57134	0.59667	0.82898	0.55946	0.55946	0.55946
		nn*	0.61205	0.61134	0.57487	0.73613	0.55970	0.55946	0.55946	0.55946
	Centers Offset	poly	0.92141	0.65032	0.59452	0.59483	0.91874	0.40323	0.44683	0.42142
		nn	**0.93186**	**0.93186**	**0.93139**	**0.93236**	**0.92950**	**0.92623**	**0.92936**	**0.92658**
		poly*	0.92043	0.92094	0.63328	0.41815	0.92086	0.44770	0.44770	0.44770
		nn*	0.72483	0.72447	0.92296	0.47662	0.72495	0.44770	0.44770	0.44770
	Label Usage	poly	**1.00**	0.23297	0.01151	0.03931	**1.00**	0.06162	0.14073	0.02023
		nn	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
		poly*	**1.00**	**1.00**	0.30232	0.45083	**1.00**	0.00	0.00	0.00
		nn*	0.50000	0.50000	0.50521	0.50000	0.00226	0.00	0.00	0.00
	Segment Overhead[%]	poly	**7.69231**	25.00	63.63636	75.51020	**7.69231**	81.25000	84.81013	83.78378
		nn	**7.69231**	**7.69231**	**7.69231**	**7.69231**	**7.69231**	**7.69231**	**7.69231**	**7.69231**
		poly*	**7.69231**	**7.69231**	29.41176	14.28571	**7.69231**	-33.3333	-9.09091	25.00
		nn*	**7.69231**	-9.09091	20.00	-9.09091	**7.69231**	-33.3333	-9.09091	25.00
Trajectory	mse	poly	0.02378	3.94293	7.68450	8.81175	0.04348	254.6476	343.2510	344.9534
		nn	0.02157	0.45997	**0.54016**	**0.47211**	0.04821	**0.61130**	**0.46287**	**0.63225**
		poly*	**0.01679**	**0.02345**	11.32192	55874.84	**0.03966**	85737.23	82544.08	58358.20
		nn*	9.74601	9.27635	3.28995	55854.09	9.74191	85737.23	82544.08	58358.20
	std	poly	0.09738	2.17017	2.38515	2.34680	0.12738	11.36520	11.69699	11.77413
		nn	0.12877	0.69356	**0.75262**	**0.70695**	0.12920	**0.80049**	**0.69195**	**0.80949**
		poly*	**0.08062**	**0.12702**	2.38548	121.5990	**0.10070**	149.0507	146.1296	123.1402
		nn*	2.21389	2.17628	1.71779	120.7779	2.19618	149.0507	146.1296	123.1402



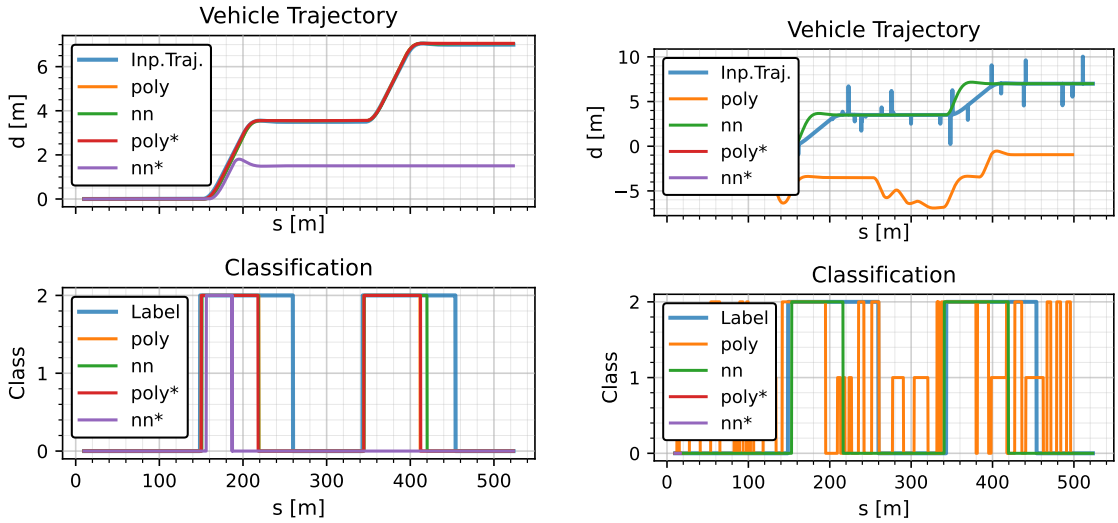
(a) DC1

(b) DC2

(c) DC3

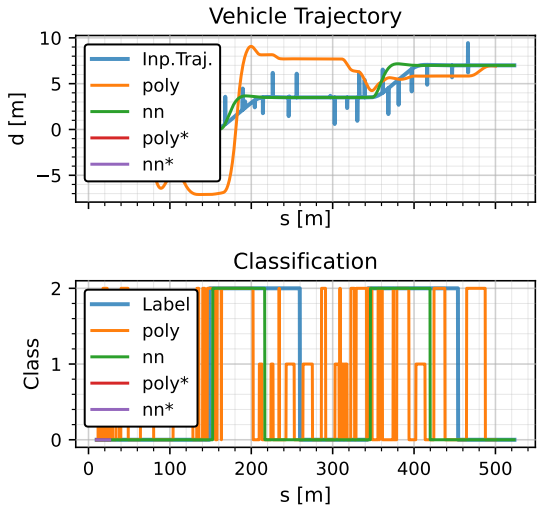
(d) DC4

Figure 5.29: Trajectory classification results for DoubleLCTC1 (DC1-DC4).

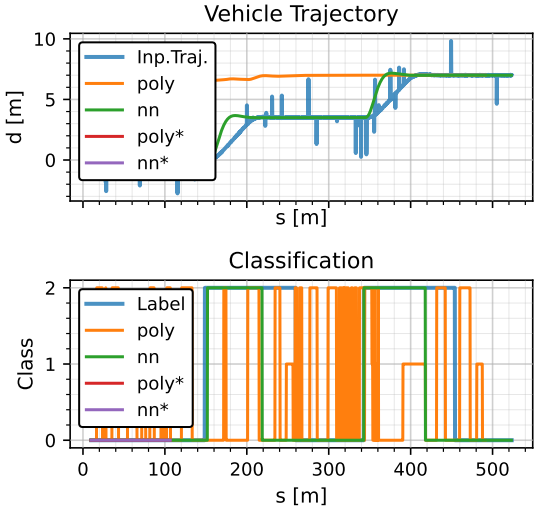


(a) DC5

(b) DC6

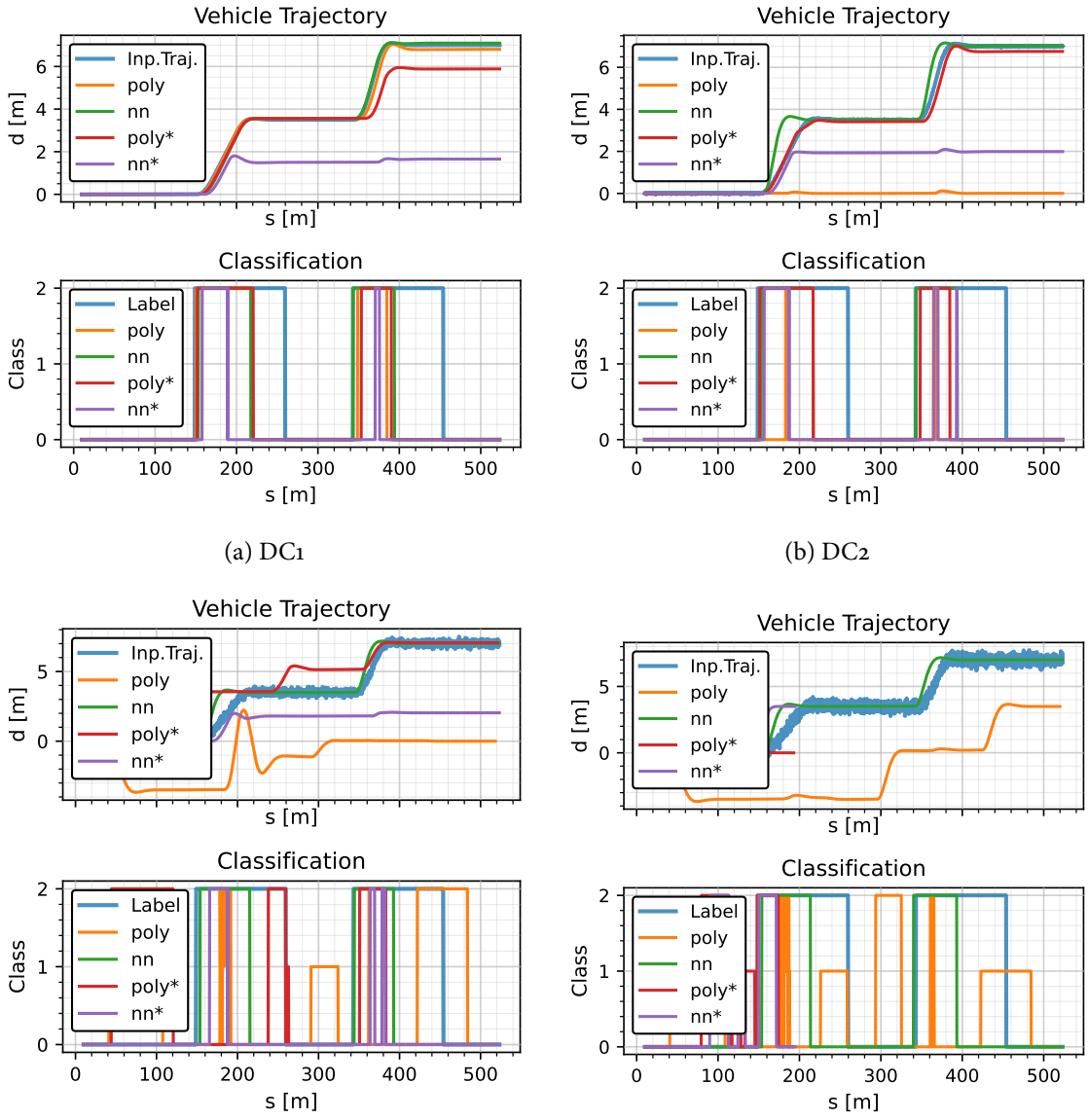


(c) DC7



(d) DC8

Figure 5.30: Trajectory classification results for DoubleLCTC1 (DC5-DC8).



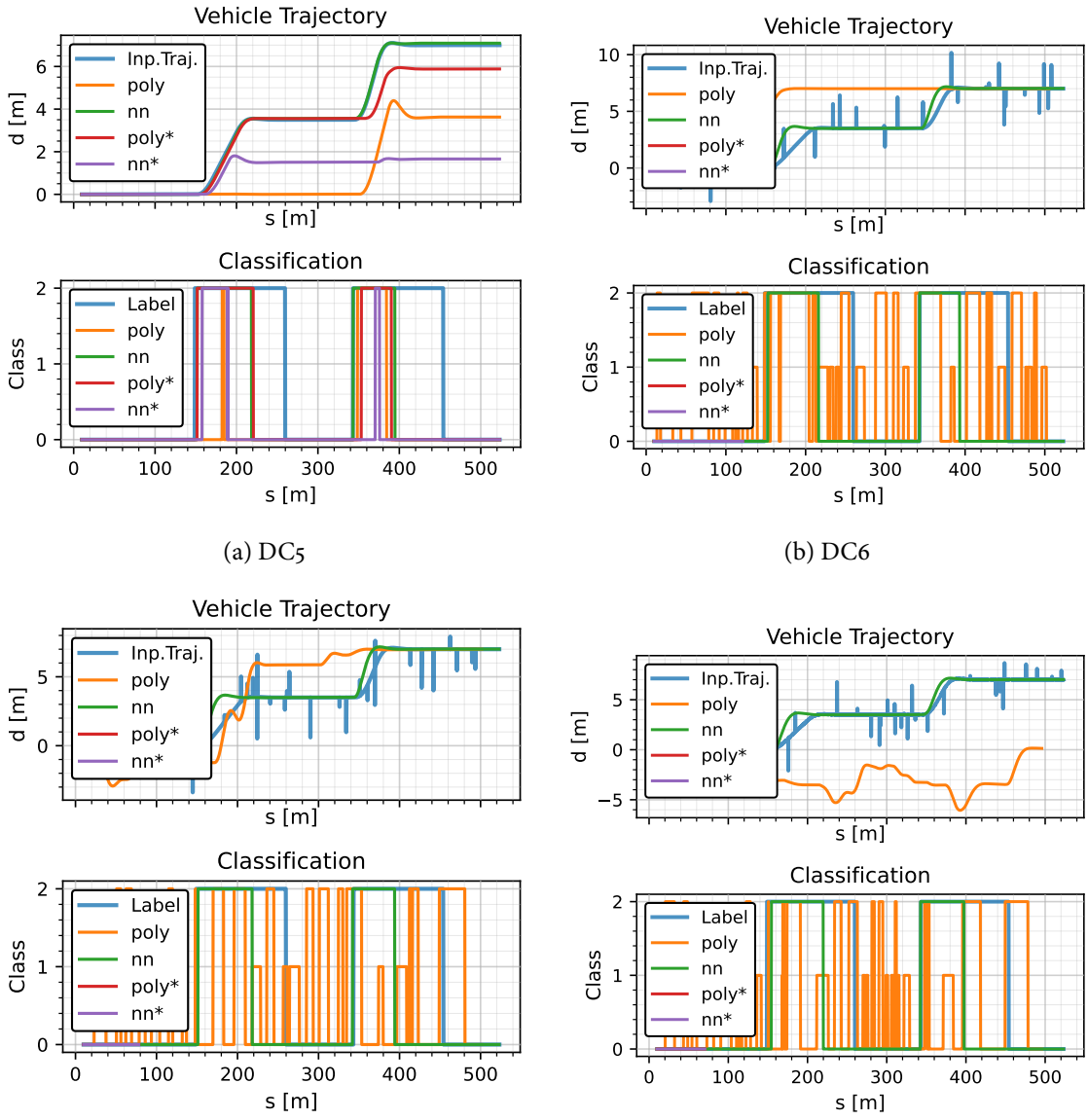
(a) DC1

(b) DC2

(c) DC3

(d) DC4

Figure 5.31: Trajectory classification results for DoubleLCTC2 (DC1-DC4).



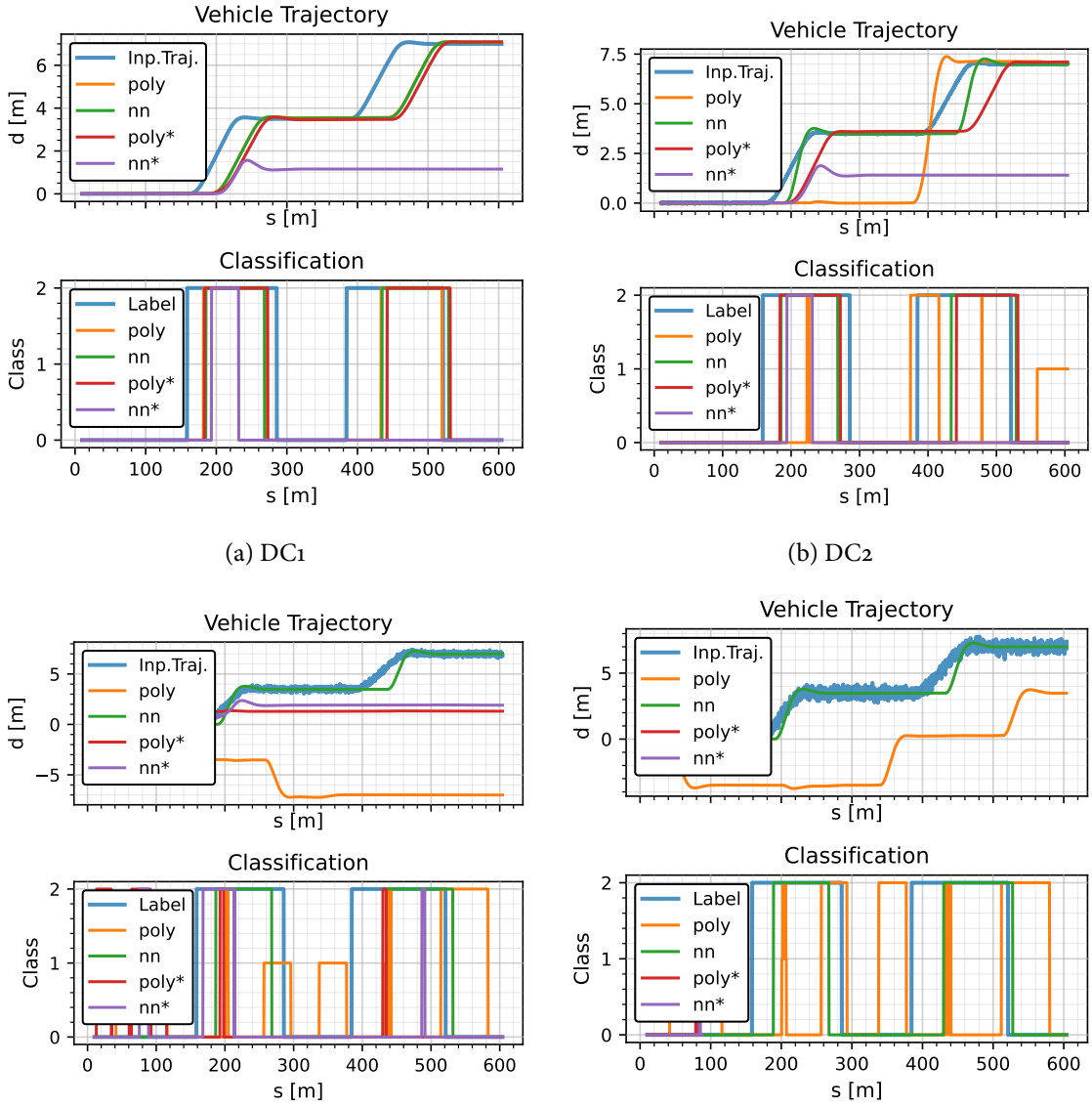
(a) DC5

(b) DC6

(c) DC7

(d) DC8

Figure 5.32: Trajectory classification results for DoubleLCTC2 (DC5-DC8).



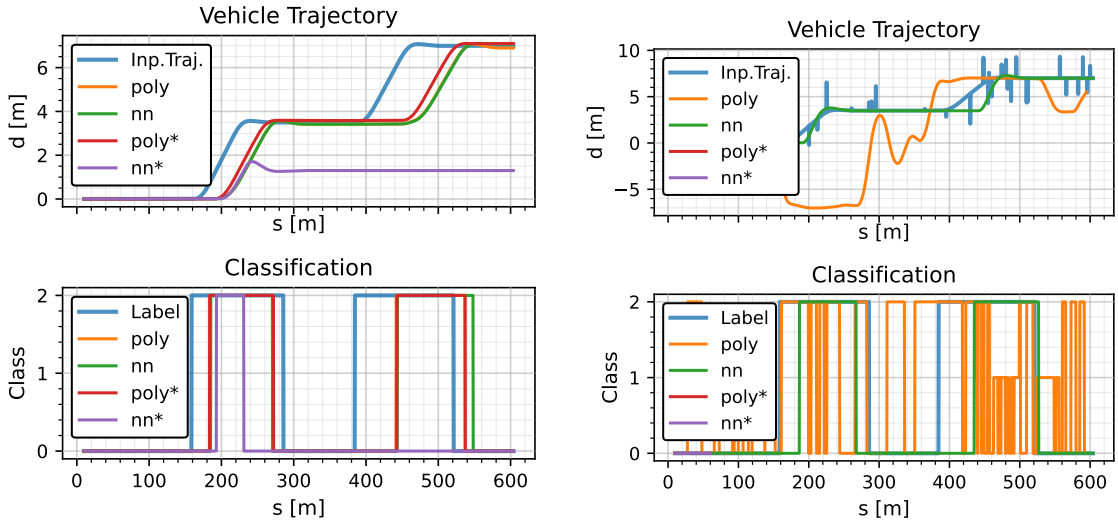
(a) DC1

(b) DC2

(c) DC3

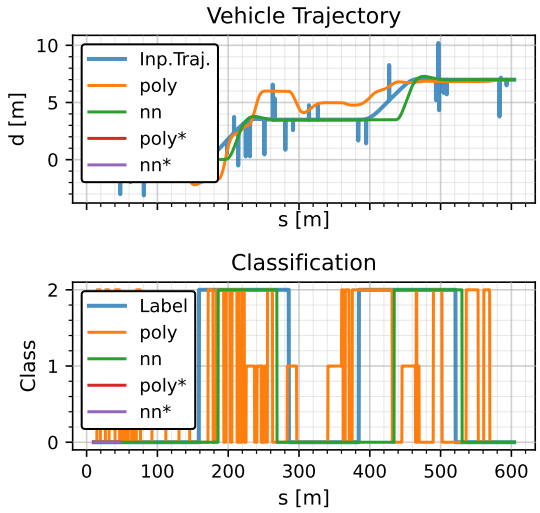
(d) DC4

Figure 5.33: Trajectory classification results for DoubleLCTC3 (DC1-DC4).

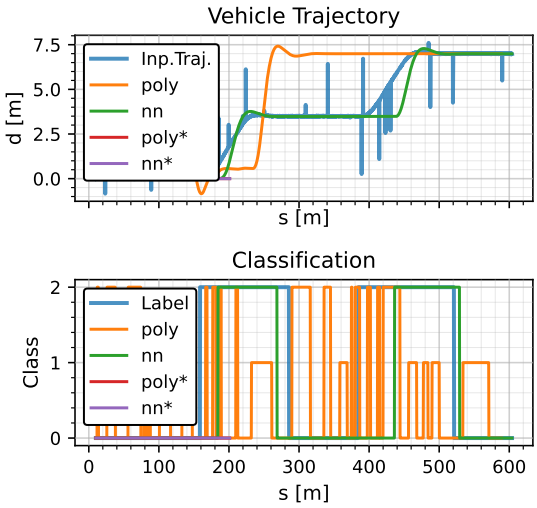


(a) DC5

(b) DC6

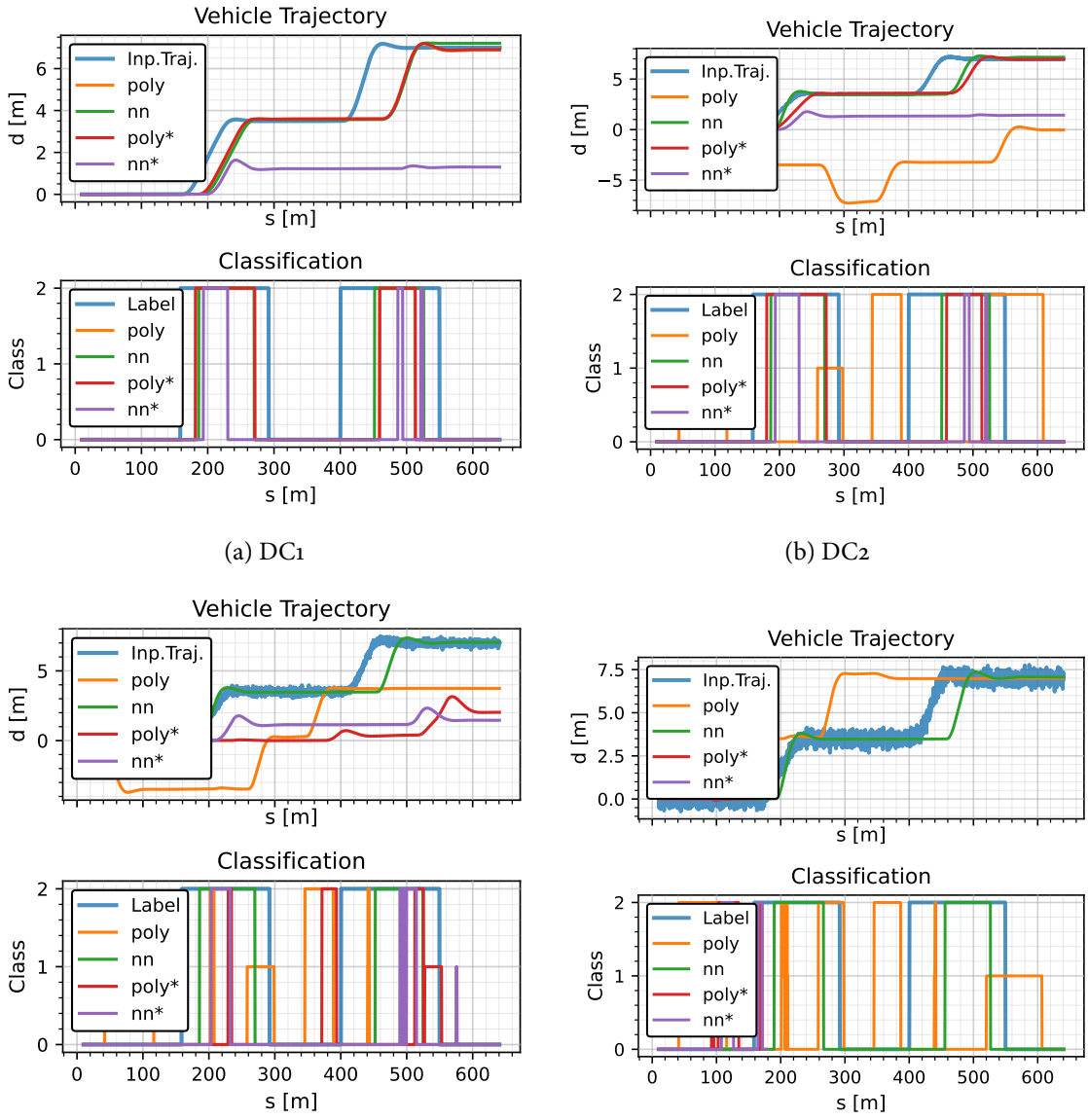


(c) DC7



(d) DC8

Figure 5.34: Trajectory classification results for DoubleLCTC3 (DC5-DC8).



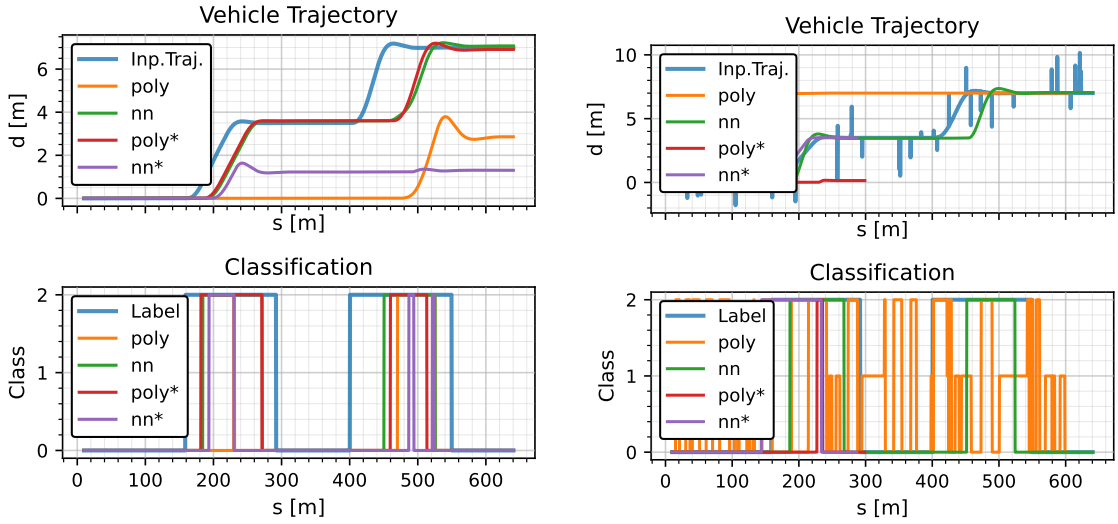
(a) DC1

(b) DC2

(c) DC3

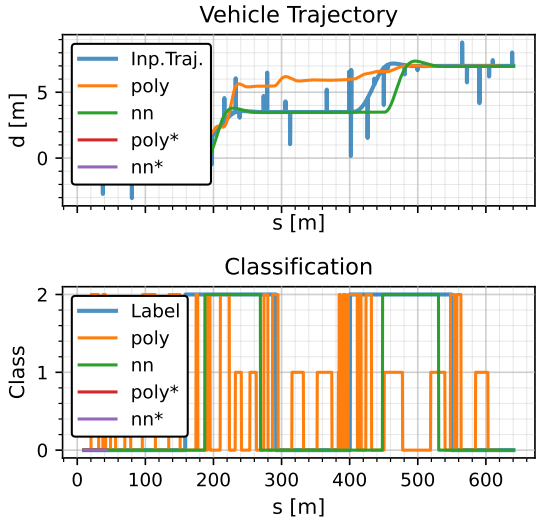
(d) DC4

Figure 5.35: Trajectory classification results for DoubleLCTC4 (DC1-DC4).

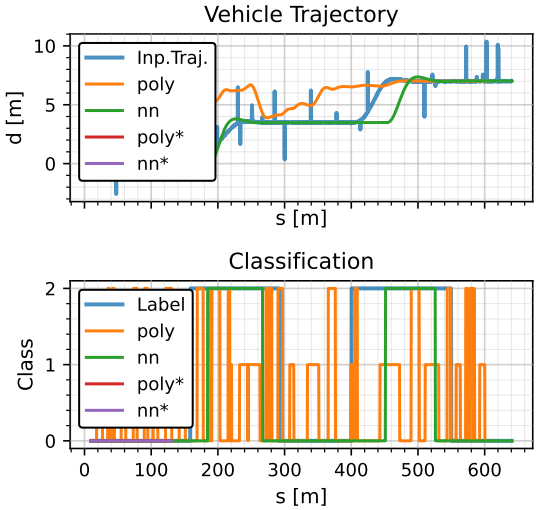


(a) DC5

(b) DC6



(c) DC7

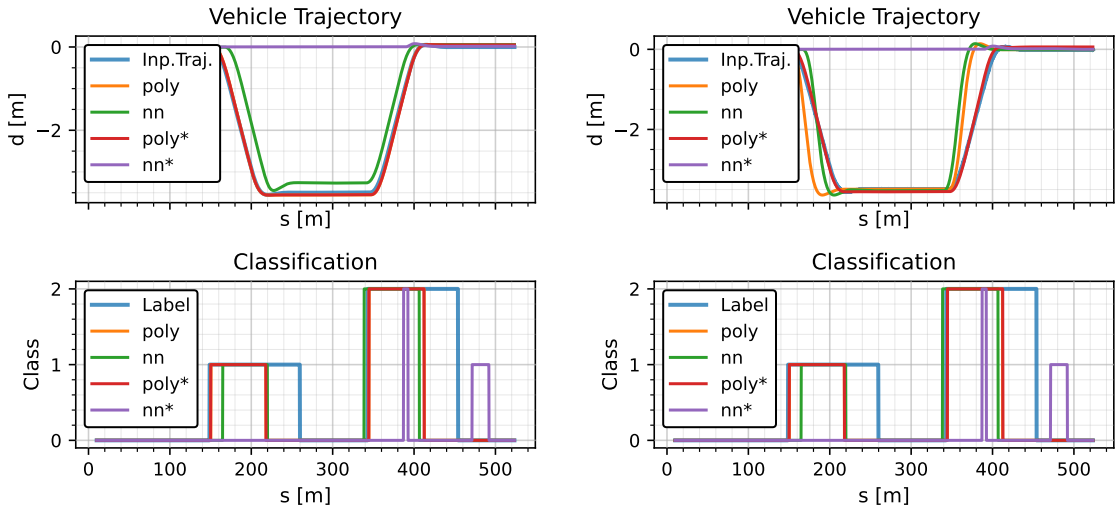


(d) DC8

Figure 5.36: Trajectory classification results for DoubleLCTC4 (DC5-DC8).

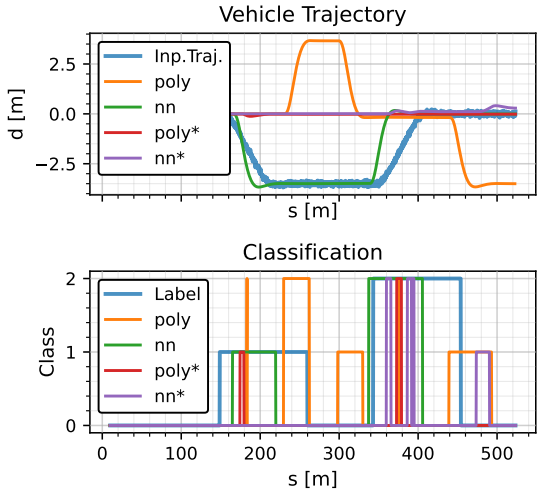
Table 5.7: Evaluation Results ushapeLC across all test cases

ushapeLC			All test case							
Distr. config:			DC1	DC2	DC3	DC4	DC5	DC6	DC7	DC8
Classification	Label Accuracy	poly	0.83166	0.83166	0.42576	0.43973	0.83088	0.34927	0.58480	0.51042
		nn	0.79043	0.79151	**0.78719**	**0.79159**	0.79448	0.78935	**0.79691**	0.79451
		poly*	**0.83209**	**0.83264**	0.58907	0.73958	**0.83209**	**0.82799**	0.59834	**0.83639**
		nn*	0.53800	0.53784	0.55922	0.50091	0.53803	0.53987	0.54106	0.52998
	Start Ends	poly	0.82850	0.82850	0.56431	0.56334	0.82603	0.56382	0.57085	0.56254
		nn	0.80685	0.80810	**0.80810**	**0.81280**	0.81185	0.80810	**0.81491**	0.81250
		poly*	**0.82898**	**0.82960**	0.58147	0.72815	**0.82898**	**0.82436**	0.59331	**0.83217**
		nn*	0.57165	0.57134	0.56227	0.56127	0.57197	0.57134	0.56540	0.60292
	Centers Offset	poly	0.92141	0.92141	0.65025	0.60039	0.92081	0.59270	0.64970	0.64876
		nn	**0.92157**	**0.92220**	**0.91970**	0.92014	**0.92155**	0.92095	0.92175	0.91541
		poly*	0.92031	0.92117	0.90137	**0.92150**	0.92086	**0.92137**	**0.92496**	**0.92243**
		nn*	0.48100	0.48116	0.48132	0.57277	0.48100	0.48100	0.47631	0.48319
	Label Usage	poly	**1.00**	**1.00**	0.01520	0.06161	**1.00**	0.03595	0.46603	0.09434
		nn	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
		poly*	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
		nn*	0.10372	0.10215	0.02133	0.06707	0.10582	0.10615	0.09259	0.50000
	Segment Overhead[%]	poly	**7.69231**	**7.69231**	50.00	62.50000	**7.69231**	52.00	29.41176	29.41176
		nn	**7.69231**	**7.69231**	**7.69231**	**7.69231**	**7.69231**	**7.69231**	**7.69231**	**7.69231**
		poly*	**7.69231**	**7.69231**	**7.69231**	**7.69231**	**7.69231**	**7.69231**	**7.69231**	**7.69231**
		nn*	**7.69231**	**7.69231**	29.41176	42.85714	**7.69231**	**7.69231**	20.00	-9.09091
Trajectory	mse	poly	0.02294	0.35497	11.09063	22.08142	0.04967	31.44155	9.55834	4.25773
		nn	0.15861	0.33260	**0.52966**	0.53415	0.17266	0.51679	**0.58593**	0.60717
		poly*	**0.01613**	**0.02065**	4.21072	**0.14280**	**0.03534**	**0.07385**	4.23886	**0.04132**
		nn*	4.38405	4.37789	4.67109	5.12952	4.33240	4.36265	4.25657	4.35726
	std	poly	0.11055	0.64447	3.30947	3.02847	0.14565	2.78276	2.96183	1.69667
		nn	0.36773	0.61233	**0.78644**	0.79156	0.36874	0.77712	**0.82192**	0.83867
		poly*	**0.09267**	**0.10681**	1.74696	**0.39468**	**0.10933**	**0.24264**	1.61020	**0.14497**
		nn*	1.82156	1.81758	1.94563	1.76799	1.78269	1.80186	1.62769	1.81249

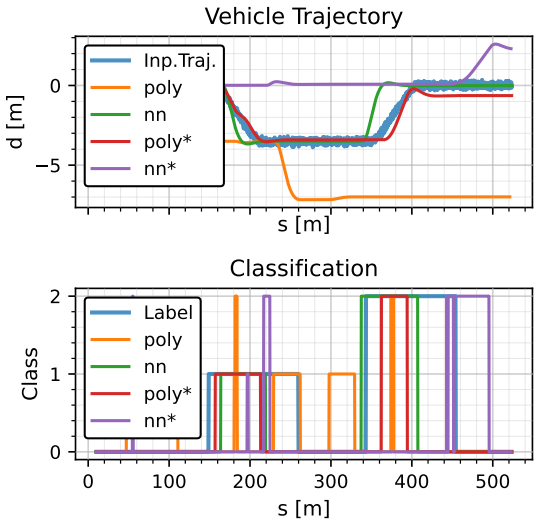


(a) DC1

(b) DC2

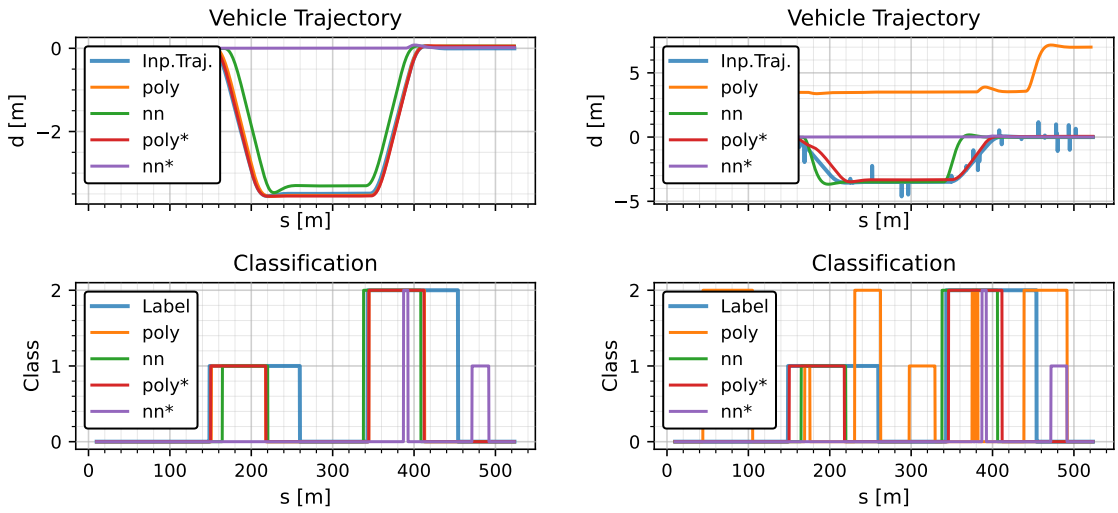


(c) DC3



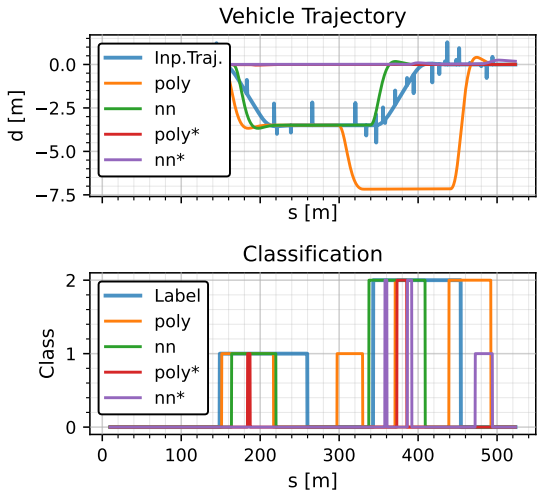
(d) DC4

Figure 5.37: Trajectory classification results for ushapeLCTC1 (DC1-DC4).

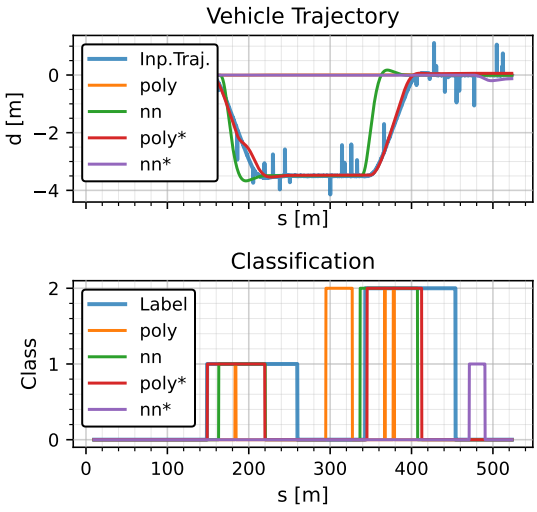


(a) DC5

(b) DC6

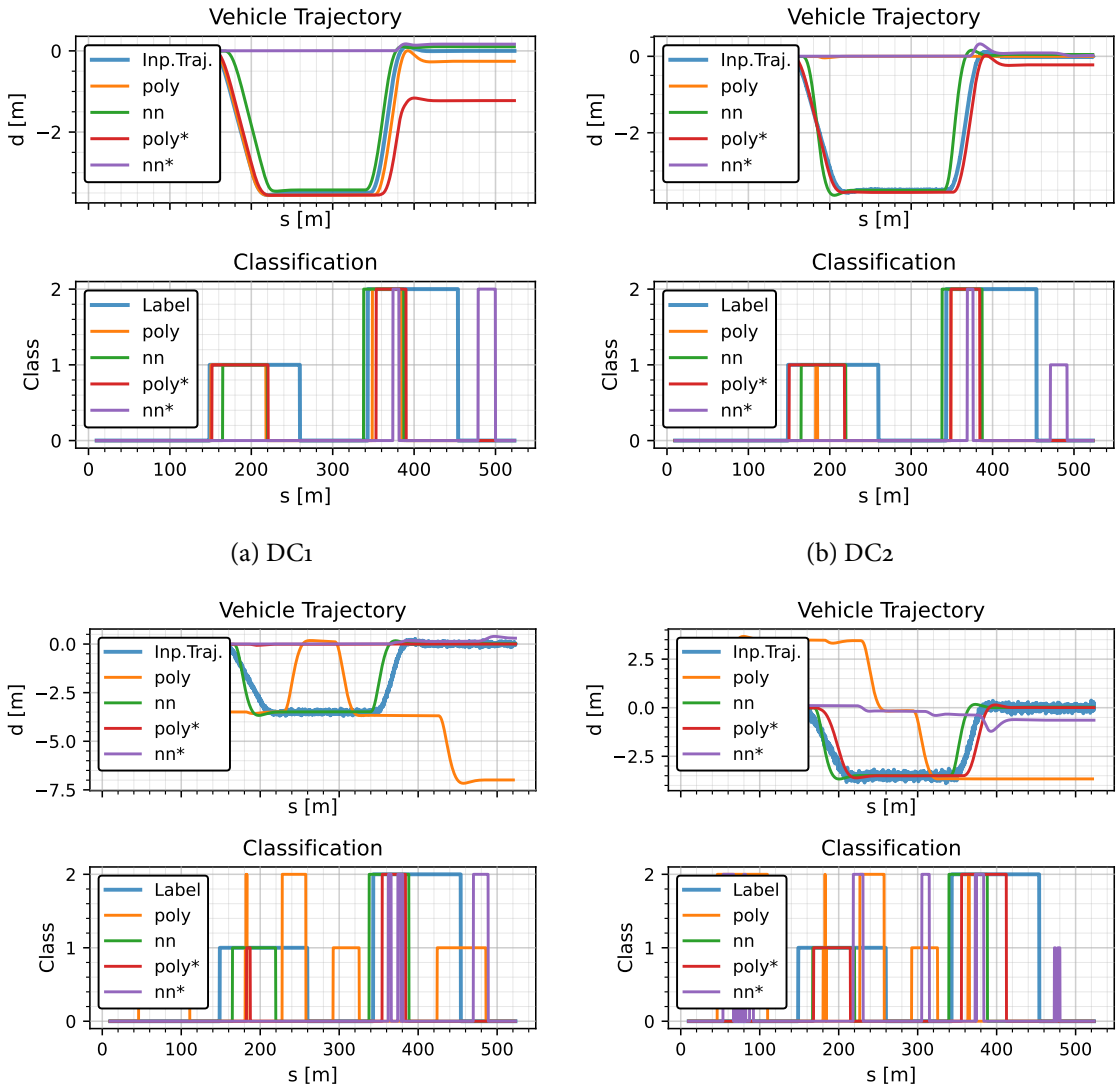


(c) DC7



(d) DC8

Figure 5.38: Trajectory classification results for ushapeLCTC1 (DC5-DC8).



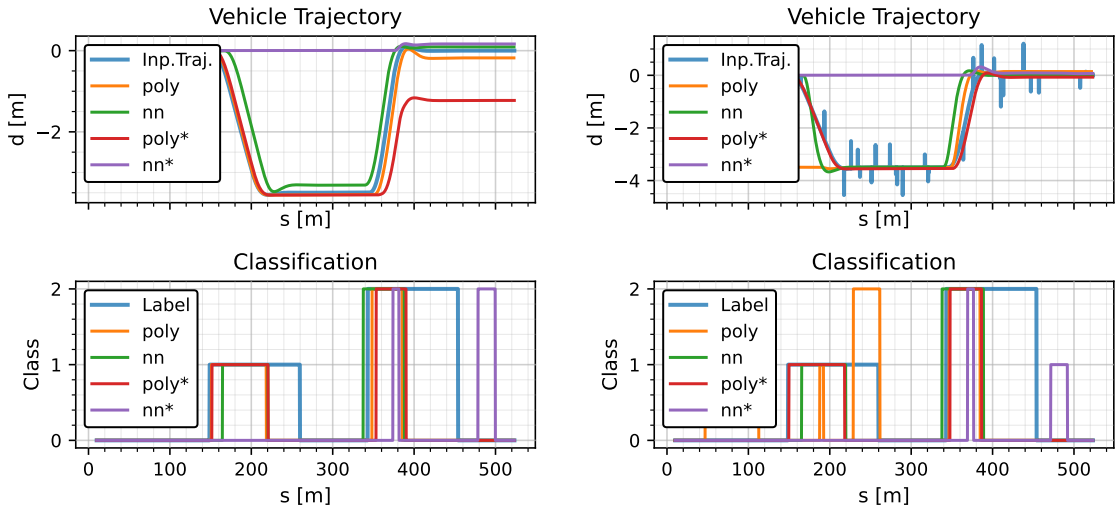
(a) DC1

(b) DC2

(c) DC3

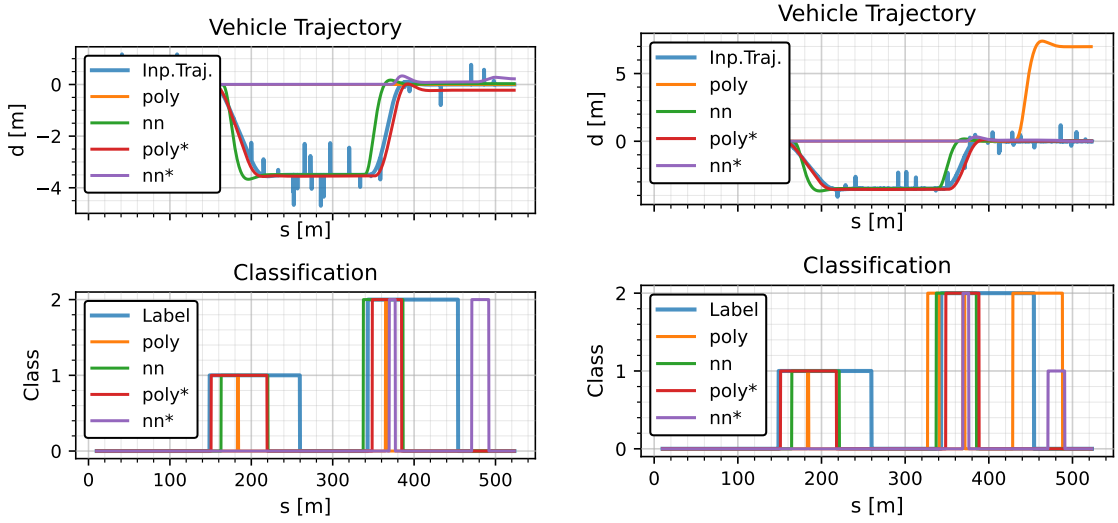
(d) DC4

Figure 5.39: Trajectory classification results for ushapeLCTC2 (DC1-DC4).



(a) DC5

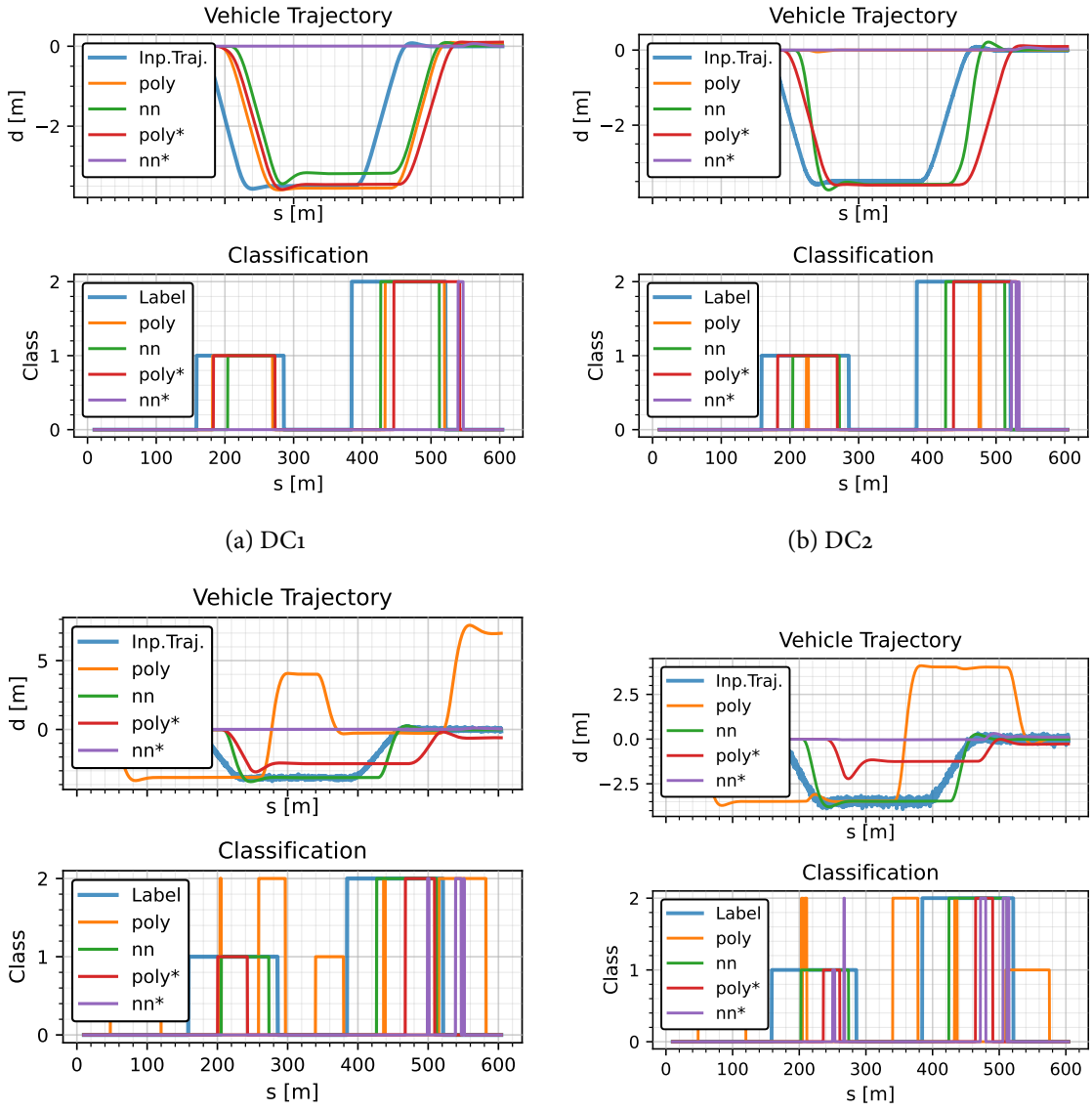
(b) DC6



(c) DC7

(d) DC8

Figure 5.40: Trajectory classification results for ushapeLCTC2 (DC5-DC8).



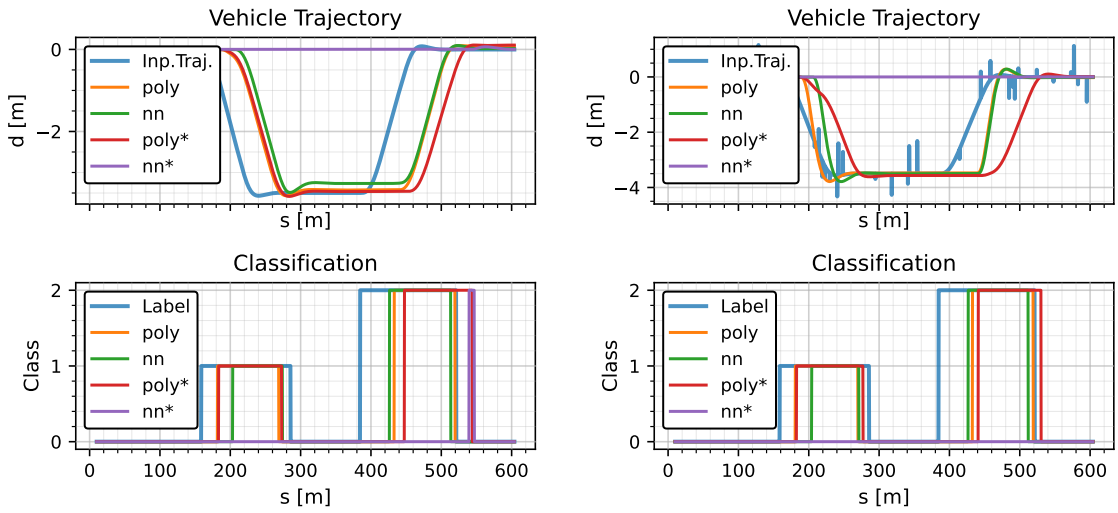
(a) DC1

(b) DC2

(c) DC3

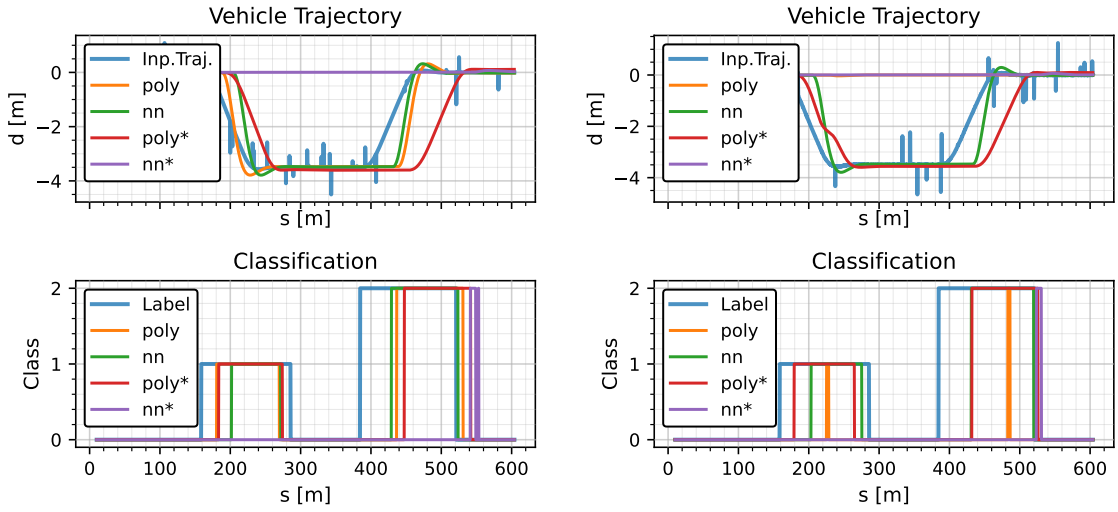
(d) DC4

Figure 5.41: Trajectory classification results for ushapeLCTC3 (DC1-DC4).



(a) DC5

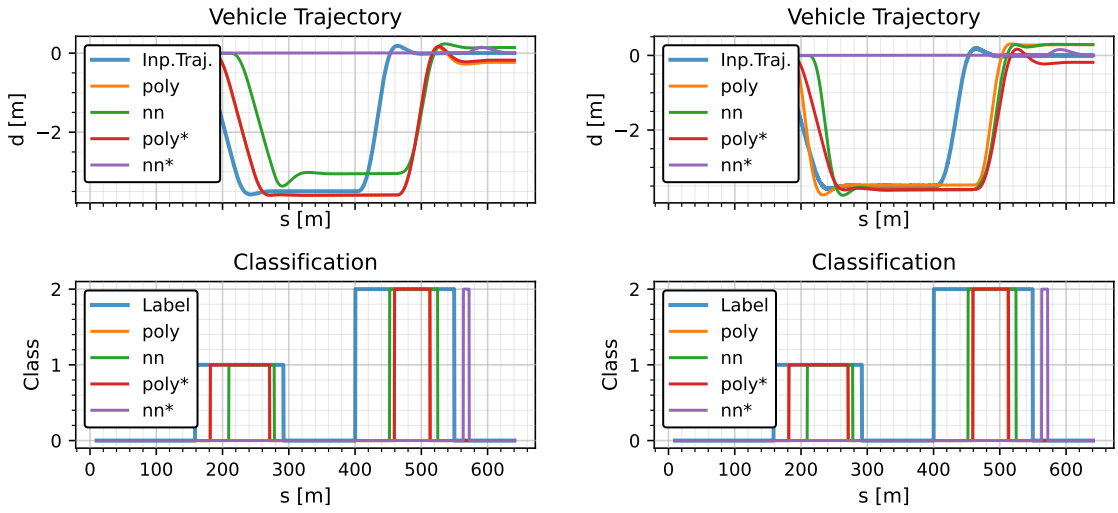
(b) DC6



(c) DC7

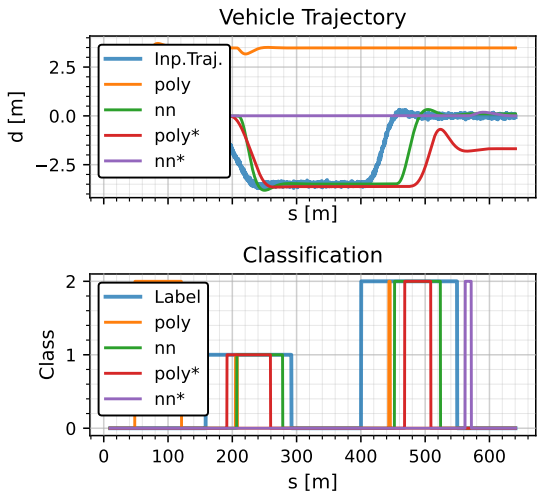
(d) DC8

Figure 5.42: Trajectory classification results for ushapeLCTC3 (DC5-DC8).

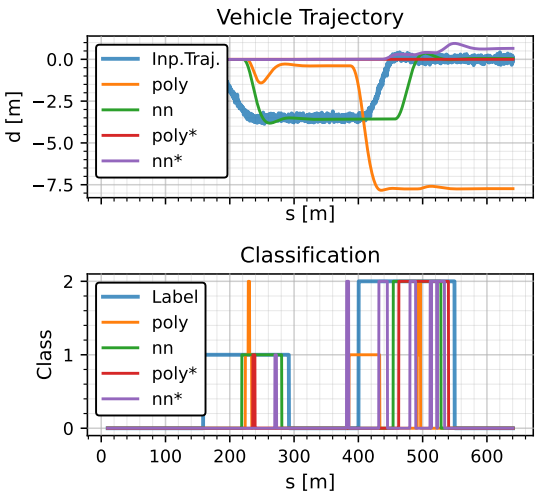


(a) DC1

(b) DC2

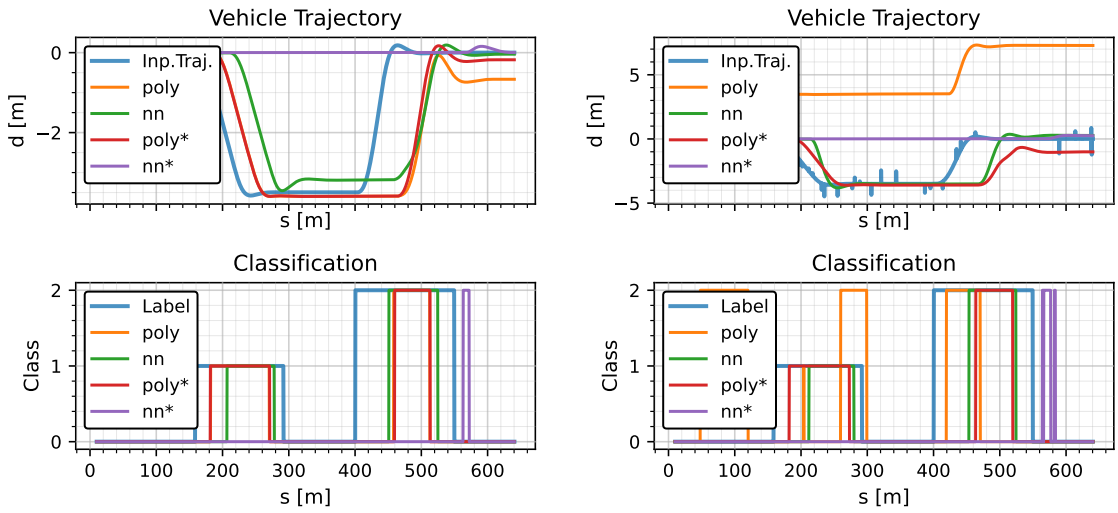


(c) DC3



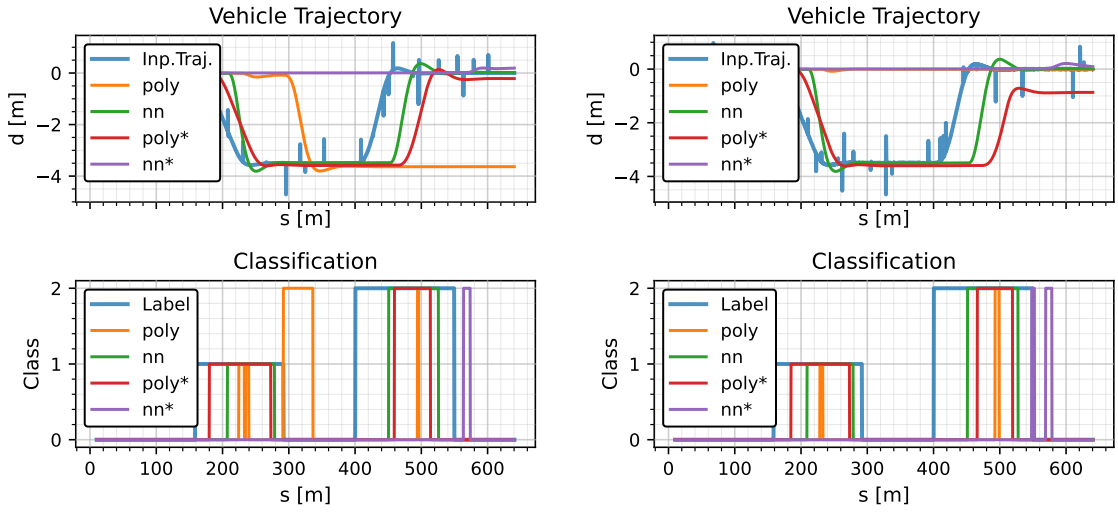
(d) DC4

Figure 5.43: Trajectory classification results for ushapeLCTC4 (DC1-DC4).



(a) DC5

(b) DC6



(c) DC7

(d) DC8

Figure 5.44: Trajectory classification results for ushapeLCTC4 (DC5-DC8).

Table 5.8: Evaluation Results Combined All across all test cases

Combined All			All test case							
Distr. config:			DC1	DC2	DC3	DC4	DC5	DC6	DC7	DC8
Classification	Label Accuracy	poly	0.87001	0.74677	0.44583	0.39294	0.79477	0.61142	0.48874	0.54302
		nn	0.84160	0.84195	**0.84015**	**0.83917**	0.83218	**0.84082**	**0.82926**	**0.84752**
		poly*	**0.87041**	**0.86828**	0.63334	0.68728	**0.87041**	0.78082	0.70127	0.77951
		nn*	0.64534	0.64510	0.66022	0.57703	0.64534	0.62621	0.62651	0.62234
	Start Ends	poly	0.86790	0.77951	0.61370	0.61714	0.78999	0.69210	0.62271	0.61388
		nn	0.84698	0.84740	**0.84748**	**0.84574**	0.83544	**0.85016**	**0.83285**	**0.85022**
		poly*	**0.86829**	**0.86612**	0.64542	0.71322	**0.86853**	0.77700	0.69681	0.77532
		nn*	0.65143	0.65125	0.63575	0.69536	0.63408	0.63411	0.63197	0.64400
	Centers Offset	poly	0.94078	0.85050	0.74278	0.72603	0.93850	0.65849	0.68774	0.68173
		nn	**0.94345**	**0.94366**	**0.94314**	**0.94314**	**0.94456**	**0.94160**	**0.94418**	**0.93860**
		poly*	0.93985	0.94055	0.83116	0.76743	0.94026	0.78263	0.78208	0.78259
		nn*	0.72766	0.72752	0.79183	0.66965	0.72770	0.63465	0.63206	0.63554
	Label Usage	poly	**1.00**	0.74432	0.01214	0.04205	**1.00**	0.36586	0.21961	0.37152
		nn	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
		poly*	**1.00**	**1.00**	0.76744	0.81694	**1.00**	0.66667	0.66667	0.66667
		nn*	0.53457	0.53405	0.47322	0.46859	0.36936	0.36872	0.36420	0.50000
	Segment Overhead[%]	poly	**8.83191**	14.60114	50.69930	68.22562	**8.83191**	48.12037	53.62952	41.43555
		nn	**8.83191**	8.83191	**8.83191**	**8.83191**	**8.83191**	**8.83191**	8.83191	**8.83191**
		poly*	14.21911	14.21911	21.45893	16.41692	14.21911	-4.84330	**8.62471**	19.98834
		nn*	14.21911	**8.62471**	29.29110	24.07592	14.21911	-4.84330	12.72727	14.39394
Trajectory	mse	poly	**0.01674**	1.53038	7.79501	22.27792	1.57271	95.51255	120.3035	117.9457
		nn	0.08844	**0.31059**	**0.43275**	**0.41486**	**0.27572**	**0.45599**	**0.38416**	**0.51465**
		poly*	1.34997	1.36405	8.52084	18629.40	1.26863	28579.11	27517.30	19454.04
		nn*	7.32032	7.12719	6.32019	18624.58	7.20453	28581.71	27518.79	19456.63
	std	poly	**0.09132**	1.12599	2.45802	3.03400	0.65794	4.94950	5.57237	5.05651
		nn	0.24908	0.56863	**0.68497**	**0.67471**	**0.36620**	**0.70173**	**0.62478**	**0.74655**
		poly*	0.46394	**0.48935**	2.30416	41.47241	0.46458	49.81354	49.68331	41.53567
		nn*	2.22894	2.20895	2.21258	41.82598	2.19820	50.81153	50.17355	42.52325

5.5.2 *Evaluation with the HighD Dataset*

Further evaluation was conducted on the *HighD* dataset [67]. The *HighD* dataset comprises top-down drone recordings at 6 German motorway sites. Recorded raw video (4K, 25 fps) was processed to extract trajectories. During this process, trajectory smoothing was applied to reduce measurement noise. In total, the *HighD* dataset includes 60 recordings totalling 16.5 h, about 110,000 tracked vehicles, approximately 45,000 km travelled, and roughly 5,600 complete lane changes. For evaluation, each vehicle trajectory was converted into a driving scenario; simulating that scenario produced a final trajectory, which was then compared with the original trajectory from the *HighD* dataset. To ensure that a lane-change manoeuvre is present in the trajectories selected for evaluation, only trajectories with a lateral offset of at least 2 m and a travelled distance greater than 390 m were selected. Additionally, to avoid evaluating performance on training data, the neural network model in Chapter 5 was trained on *HighD* recordings 0–12, while evaluation used recordings 50–60. This resulted in 2551 trajectories used for evaluation.

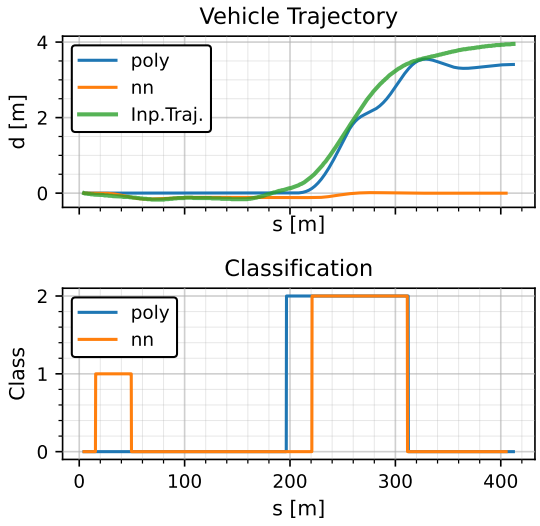
Road lane lines required for the classification methodology in this thesis were derived from the recording metadata accompanying each *HighD* scene. The metadata provide lateral positions of the upper and lower road lane lines; these were parsed and ordered to obtain candidate line sets for each lane. The resulting constant offset was then applied uniformly to all lines. After all recordings had been processed, duplicate entries were removed. The positions were finally consolidated into the lower and upper road lane line sets.

Both polynomial and neural network methods for lateral classification were evaluated on the same trajectories. Longitudinal classification was performed, as before, using greedy knot reduction. Both the polynomial and neural network classification methods were combined separately with the longitudinal classification, and two separate logic driving scenarios were created for each *HighD* trajectory selected for evaluation. Neither simulation-based filtering nor any other type of filtering was applied to the original *HighD* trajectory data.

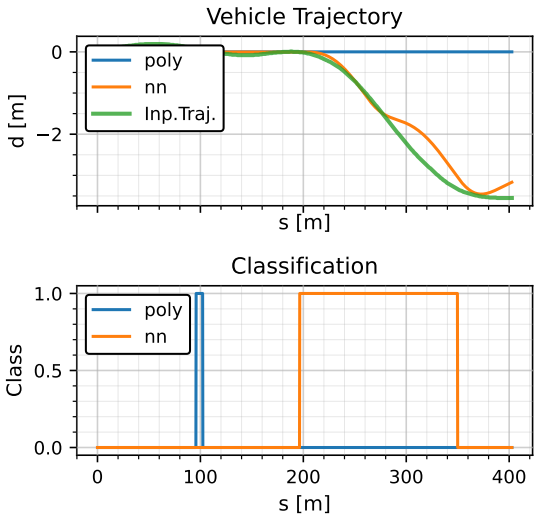
Results are shown in Table 5.9. The values presented are the mean squared error (MSE) between the trajectory obtained by simulating the logical scenario and the original trajectory from *HighD*. Simulations were performed in dSPACE ASM [96]. The aforementioned table reports the best and worst MSE values for the single case, top 1%, and top 10% for the polynomial and neural-network methods used for lateral classification. “Poly focus” and “nn focus” rank trajectories by each method separately, while “combined” selects trajectories that are simultaneously best (or worst) for both methods. Best and worst cases for each column can be seen in Figure 5.45. Figure 5.46 shows the vehicle states for each method for the combined best MSE.

Table 5.9: HighD dataset MSE/STD results for poly and nn lateral classification methods

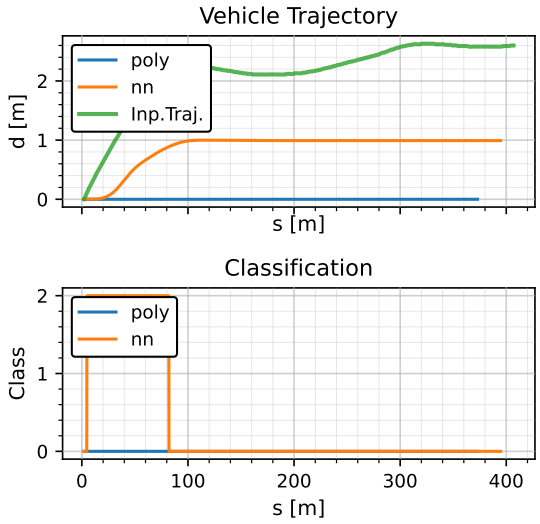
		Poly ranking		NN ranking		Joint ranking	
Desc.	Method	MSE	STD	MSE	STD	MSE	STD
Best	Poly	**0.13049**	-	23.56273	-	**0.18260**	-
	NN	3.14950	-	**0.13286**	-	0.21392	-
Best 1%	Poly	**0.30068**	0.06927	2.58584	5.15805	0.37916	0.10406
	NN	5.64835	6.64199	**0.21839**	0.03800	**0.37652**	0.09279
Best 10%	Poly	**0.74809**	0.26982	2.58168	4.71538	0.94123	0.40355
	NN	4.06431	4.35679	**0.46415**	0.13440	**0.86220**	0.39127
Worst	Poly	**301.51767**	-	41.95955	-	238.58242	-
	NN	238.58242	-	**224.54304**	-	**224.54304**	-
Worst 1%	Poly	**91.41882**	59.07759	70.39401	51.64166	85.14492	62.73221
	NN	69.53912	50.64500	**80.67151**	44.25280	**76.67098**	47.06415
Worst 10%	Poly	**32.43088**	28.13027	21.20336	26.47170	28.49086	29.41993
	NN	23.80357	31.15034	**30.29440**	23.22376	**26.65706**	24.34943
Total Sum	Poly	8.85296	12.61013				
	NN	6.86614	11.33649				
Selection	best	5.26955	9.61943				
	worst	10.44955	13.54827				



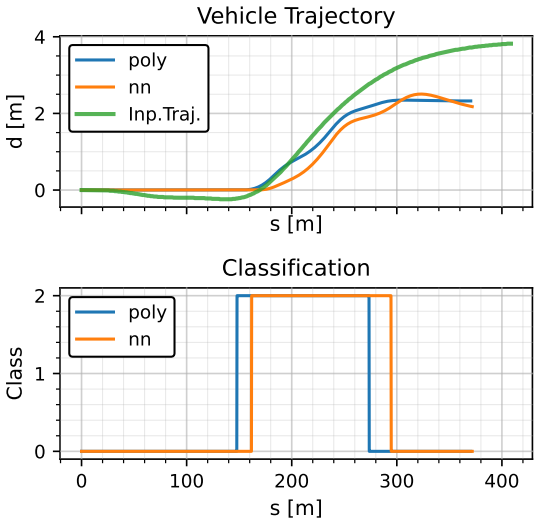
(a) The best MSE focus on poly



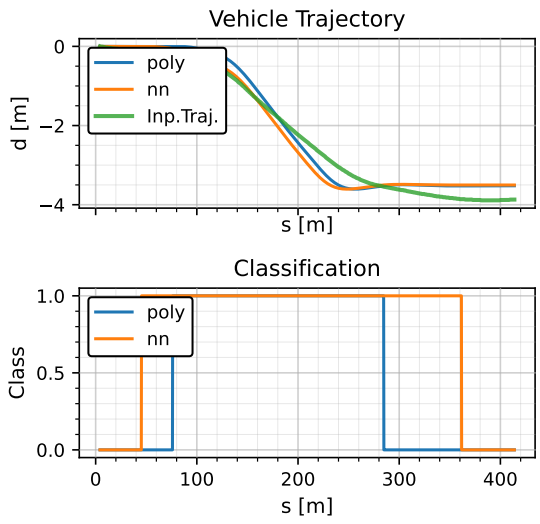
(b) The best MSE focus on nn



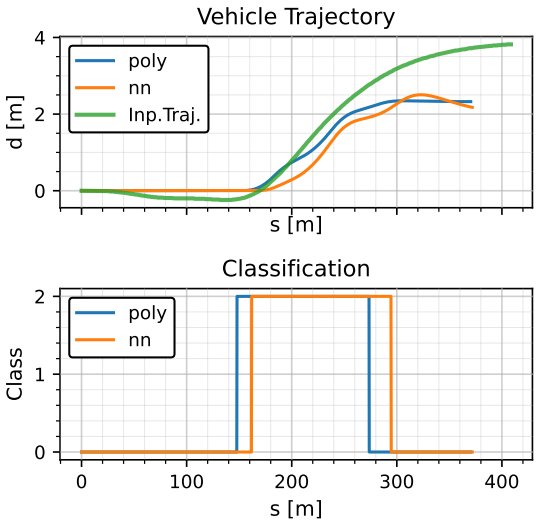
(c) The worse MSE focus on poly



(d) The worse MSE focus on nn



(e) The best MSE scenairon



(f) The worse MSE scenairon

Figure 5.45: HighD results best and worse based on MSE

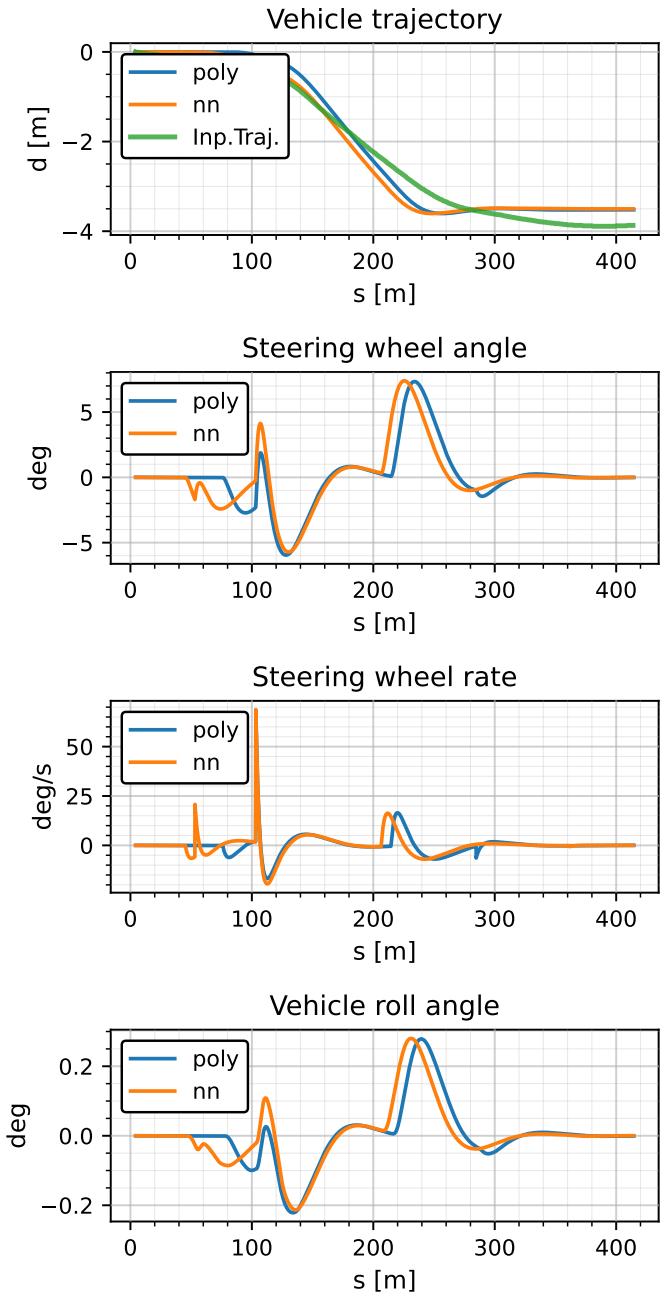


Figure 5.46: Vehicle state from simulation

5.6 SUMMARY

This chapter presented a complete workflow for converting recorded vehicle trajectories into simulator-ready logic scenarios. Vehicle actions were classified in both lateral and longitudinal dimensions using a unified curvilinear representation (S, D, T) and road lane-line trajectories R. Lateral actions were classified into lane keeping, lane change left, and lane change right, while longitudinal actions were segmented into constant-acceleration phases representing acceleration, braking, and constant velocity cruising. Two complementary approaches were introduced for lateral classification. The polynomial method detects lane-line crossings using continuous polynomial families for both the vehicle trajectory and lane lines, enabling sub-sample crossing detection and estimating lane-change start and end points from the inflection points of a fitted third-degree model. The neural-network approach was developed through an architecture study and then refined into an extended model combining convolutional feature extraction with bidirectional LSTM layers. To enable learning and fair comparison, a combined dataset was constructed from HighD, NGSIM, and the synthetic HLCsim dataset, augmented by mirroring and multiple noise levels. Longitudinal classification was implemented using greedy knot reduction on quadratic splines, producing a compact set of constant-acceleration segments that are directly interpretable as longitudinal actions. Lateral and longitudinal classifications were then cross-paired and merged into logic segments using an XOR-style boundary rule, ensuring that any change in either action type starts a new segment. Initial scenario conditions (lane, offset, position, and velocity) were derived from the first trajectory samples and the first segment parameters, enabling direct scenario instantiation.

Finally, the full conversion process was evaluated at the scenario level using both synthetic reference logic scenarios with controlled distortions and real-world HighD trajectories. Results showed that neural-network methods achieved the best overall lateral classification performance across noise levels, while the polynomial method remained competitive on low-noise data and offered continuous, centred detections. The chapter, therefore, establishes the methodological foundation for subsequent chapters, where these classification outputs are used for complete logic scenario reconstruction and simulation-based analysis.

6

Conclusion and Outlook

This thesis presents a complete approach for converting recorded vehicle trajectories into logic scenarios. The main goal was to bridge the gap between raw trajectory data and structured logic scenarios that can be systematically modified and reused for scenario-based testing. To achieve this, the work combined vehicle action classification, logic segment formation, simulation-based filtering using an extended-horizon MPC, and novel evaluation metrics. A key contribution is the development of methods for lateral classification of vehicle actions. Two complementary lateral classification methods were implemented and evaluated: a polynomial method based on approximation using polynomial functions, and a neural network method based on convolutional feature extraction combined with bidirectional recurrent layers. The polynomial method provides sub-sample lane line crossing detection by treating both road lane lines and vehicle trajectories as polynomial functions. The neural network approach provides higher classification performance and improved noise robustness when trained on a diverse dataset and evaluated using consistent input representations. For longitudinal actions, a greedy knot reduction method was applied to obtain constant-acceleration segments. This representation yields interpretable motion primitives and produces a description of the longitudinal actions of a vehicle. The thesis further defined how lateral and longitudinal classifications are used to form logic segments and assemble them into logic scenarios. Logic segments are defined whenever either the lateral or longitudinal action changes, ensuring that each segment corresponds to a specific vehicle motion. A practical method for extracting scenario initial conditions was provided, enabling full definition of logic scenarios. The thesis also presents a simulation-based filtering method based on a modified MPC with an extended horizon. The proposed approach reconstructs a physically consistent trajectory by simulating the motion of a vehicle model through corrupted measurements. This design improves stability in sequential goals and reduces the impact of noise and outliers in a manner consistent with the simulated vehicle dynamics model. The evaluation showed that filtering input trajectories is particularly beneficial when they are distorted by measurement noise and outlier artefacts. In addition to trajectory reconstruction, the same simulation-based filtering method was shown to support vehicle parameter estimation, with a particular focus on vehicle mass. By evaluating the consistency between simulated and recorded trajectories under varying mass assumptions, the extended-horizon MPC formulation enables indirect mass estimation without requiring additional sensors or force measurements. Although the achievable accuracy is strongly dependent on trajectory excitation and manoeuvre richness,

the results demonstrate that the proposed approach can be used to extract physically meaningful vehicle parameters from recorded trajectory data. To properly assess lane change classification in the context of logic scenario generation, this thesis introduced novel metrics that complement accuracy and confusion matrix measures. The proposed metrics quantify properties that are critical for scenario reconstruction but are not captured by standard classification scores: alignment of lane change start and end points, positional alignment of the manoeuvre centre, and the practical usability of detected segments when false positives occur. These metrics provide more informative feedback about the quality of lateral vehicle action classification results and allow easier comparison of methods whose errors may have similar accuracy but very different impact on logic scenario formation. The complete conversion process was evaluated at the scenario level using both synthetic reference logic scenarios and real-world trajectories from the HighD dataset. Synthetic scenarios enabled controlled experiments with predefined ground truth and systematic distortion configurations. Evaluation on HighD data demonstrated applicability to real-world trajectories and highlighted the practical constraints of datasets that do not natively contain lane change labels. Across experiments, neural-network-based lateral classification achieved the best overall performance, particularly under increased noise levels. The polynomial method showed advantages in continuity and centred detections under low-noise conditions, and it remains a useful fallback when the neural network does not provide usable results.

Overall, the results confirm that the work presented in this thesis can convert vehicle trajectories into modifiable logic scenarios, thereby, enabling additional testing value to be extracted from recorded trajectories.

Several limitations remain. First, the performance of learning-based classifiers depends on the representativeness of the training data, and domain shifts between data sources can reduce generalisation. Second, polynomial fitting and lane line crossing detection can be computationally demanding, although this is acceptable for offline processing. Third, manual labelling was required for parts of the real-world dataset, introducing label uncertainty that bounds the achievable accuracy of supervised learning and metric interpretation. Finally, the scenario reconstruction quality is influenced by the selected vehicle model, which can introduce mismatch not attributable to classification alone.

Future work can extend this thesis in several directions. Domain adaptation and self-supervised learning could reduce reliance on manual labels and improve robustness across datasets and sensor modalities. Finally, logic scenario representations could be expanded beyond time-based segment termination by incorporating event-based conditions and interaction constraints, enabling multi-agent scenario reconstruction and more realistic evaluation of complex traffic situations.

BIBLIOGRAPHY

[1] Nidhi Kalra and Susan M. Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? Technical Report RR-1478, RAND Corporation, 2016.

[2] Intelligent transport systems — adaptive cruise control systems — performance requirements and test procedures, 2002.

[3] Transport information and control systems — forward vehicle collision warning systems — performance requirements and test procedures, 2002.

[4] Intelligent transport systems — lane departure warning systems — performance requirements and test procedures, 2007.

[5] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer. Defining and substantiating the terms scene, situation, and scenario for automated driving. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 982–988, 2015.

[6] Till Menzel, Gerrit Bagschik, and Markus Maurer. Scenarios for development, test and validation of automated vehicles, 2018.

[7] Vjekoslav Diklić and Ivan Marković. Lateral classification of vehicle behavior for automated logic scenario generation. *Automatika*, 66(4):625–637, 2025.

[8] S. Zhao, Y. Wu, X. Zhu, B. Zhou, H. Bao, and L. Zhang. Research on automatic classification for driving scenarios based on big data and ontology. In *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, pages 1857–1862, 2018.

[9] Gerrit Bagschik, Till Menzel, and Markus Maurer. Ontology based scene creation for the development of automated vehicles. 06 2018.

[10] Till Menzel, Gerrit Bagschik, Leon Isensee, Andre Schomburg, and Markus Maurer. From functional to logical scenarios: Detailing a keyword-based scenario description for execution in a simulation environment. pages 2383–2390, 06 2019.

[11] ISO. *26262: Road Vehicles : Functional Safety*. Geneva, Switzerland, 2016.

[12] Yuanqi Cai, Bo Liu, Yue Lu, and Lei Zhang. Data-driven scenario generation for testing automated vehicles: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):10039–10055, 2022.

[13] Fabian Batsch, Marcel Strohbeck, and Matthias Althoff. Data-driven generation of roundabout scenarios for testing and validation of autonomous vehicles. In *2023 IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 1181–1188, 2023.

[14] Constantin Müller, Florian Laumann, Tobias Schamm, and Matthias Althoff. DriveSceneGen: Data-driven scenario generation for simulation-based testing of automated vehicles. In *2023 IEEE Intelligent Vehicles Symposium (IV)*, pages 1272–1279, 2023.

[15] Minh Nguyen, H. Eric Tseng, Dimitar Filev, and Ilya Kolmanovsky. Avastra: Adaptive stress testing of autonomous vehicles using state and trajectory augmentation. *IEEE Transactions on Intelligent Vehicles*, 9(1):134–146, 2024.

[16] Matthias Althoff, Maximilian Koschi, Fabian Batsch, Marcel Strohbeck, et al. Commonroad scenario factory 2.0: A modular framework for data-driven scenario generation and variation. *IEEE Transactions on Intelligent Transportation Systems*, 2025. early access.

[17] Philip Polack, Florent Altche, Brigitte d'Andrea Novel, and Arnaud de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, June 2017.

[18] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a Frenet Frame. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 987–993, Anchorage, AK, USA, 2010. IEEE.

[19] Raj Rajamani. *Vehicle Dynamics and Control*. Springer, Boston, MA, 2nd edition, 2012.

[20] Matthias Althoff, Markus Koschi, and Stefanie Manzinger. Commonroad: Composable benchmarks for motion planning on roads. In *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017.

[21] Maximilian Kloock, Christopher Mutschler, Stefan Lüke, and Jürgen Beyerer. Trajectory-to-map matching via multi-stage dynamic time warping. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 1420–1427, 2022.

[22] Julius Ziegler, Philipp Bender, Thao Dang, and Christoph Stiller. Trajectory planning for bertha — A local, continuous method. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 450–457, Dearborn, MI, USA, 2014. IEEE.

[23] Fei Lai and Chaoqun Huang. Seventh-degree polynomial-based single lane change trajectory planning and four-wheel steering model predictive tracking control for intelligent vehicles. *Vehicles*, 6(4):2228–2250, 2024.

[24] Dongxue Zhang, Xiaohong Jiao, and Ting Zhang. Lane-changing and overtaking trajectory planning for autonomous vehicles with multi-performance optimization

considering static and dynamic obstacles. *Robotics and Autonomous Systems*, 182:104797, 2024.

[25] Carl de Boor. *A Practical Guide to Splines*, volume 27 of *Applied Mathematical Sciences*. Springer, 1978.

[26] Carl de Boor and G. A. Watson. Good approximation by splines with variable knots II. In *Conference on the Numerical Solution of Differential Equations*, pages 12–20, Berlin, Heidelberg, 1974. Springer.

[27] Wayne Tiller. Knot-removal algorithms for nurbs curves and surfaces. *The Visual Computer*, 9(3):135–147, 1992.

[28] David Ruppert. Selecting the number of knots for penalized splines. *Journal of Computational and Graphical Statistics*, 11(4):735–757, 2002.

[29] Mark H Hansen and Charles Kooperberg. Spline adaptation in extended linear models. *Statistical Science*, 17(1):2–51, 2002.

[30] T. C. M. Lee. On algorithms for ordinary least squares regression splines: A comparative study. *Journal of Statistical Computation and Simulation*, 73(8):643–658, 2003.

[31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[32] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[33] Alan M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[34] Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[35] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[36] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[37] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition, 2018.

[38] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, NY, 2nd edition, 2009.

[39] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[40] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[41] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, July 2019.

[42] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[43] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[44] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[45] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[46] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[47] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 2047–2052, 2005.

[48] He Zhang, Zhixiong Nan, Tao Yang, Yifan Liu, and Nanning Zheng. A driving behavior recognition model with Bi-LSTM and multi-scale CNN. *Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 284–289, 2020.

[49] Hafiz Ashfaq, S. Maqbool, M. A. Khan, M. Rahman, I. Mehmood, and H. Song. Dual attention Bi-LSTM based approach for lane change prediction in autonomous vehicles. *Applied Sciences*, 13(17):7930, 2023.

[50] Python Software Foundation. Python: A programming language. https://www.python.org/, 2023. Version 3.x, accessed 2025-09-03.

[51] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[52] J. Richalet, A. Rault, J.L. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14(5):413–428, 1978.

[53] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, June 2000.

[54] Brian Paden, Michal Cap, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, March 2016.

[55] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat . MPC-based approach to active steering for autonomous vehicle systems. *International Journal of Vehicle Autonomous Systems*, 3(2-4):265–291, 2005.

[56] Adam Domina and Viktor Tihanyi. LTV-MPC approach for automated vehicle path following at the limit of handling. *Sensors*, 22(15), 2022.

[57] Jimmy Z. Y. Lu, Ashraf Abualfellat, and Reza Zarringhalam. Achieving automated vehicle path following with blend path curvature control. In *2022 American Control Conference (ACC)*, pages 3158–3163, 2022.

[58] H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, 1965.

[59] Vjekoslav Diklić and Ivan Marković. Vehicle trajectory filtering based on optimal control and simulator-based reconstruction. In *2025 MIPRO 48th ICT and Electronics Convention*, pages 1985–1990, 2025.

[60] Tomas Baca, Daniel Hert, Giuseppe Loianno, Martin Saska, and Vijay Kumar. Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 6753–6760. IEEE Press, 2018.

[61] Ana Batinovic, Jurica Goricanec, Lovro Markovic, and Stjepan Bogdan. Path planning with potential field-based obstacle avoidance in a 3d environment by an unmanned aerial vehicle. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 394–401, 2022.

[62] H. Pacejka and I.J.M. Besselink. *Tire and Vehicle Dynamics*. Electronic publications. Elsevier Science, 2012.

[63] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[64] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, April 2005.

[65] Vjekoslav Diklić and Branimir Novoselnik. Comparison of linear and nonlinear model predictive control for vehicle path following. In *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*, pages 1794–1799, 2024.

[66] Products - dspace. https://www.dspace.com/en/inc/home/products, 2021. (Accessed on 01/26/2021).

[67] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. The HighD dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2118–2125, 2018.

[68] Yi He, Bo Cao, and Ching-Yao Chan. WUT-NGSIM: A high-precision and trustworthy vehicle trajectory dataset, 2022.

[69] Vjekoslav Diklić. HLCsim: Highway lane change from simulation, 2023. Dataset.

[70] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[71] Vjekoslav Diklić. LC scenarios for lane change classification, 2025. Dataset.

[72] A. Sonka, F. Krauns, R. Henze, F. Küçükay, R. Katz, and U. Lages. Dual approach for maneuver classification in vehicle environment data. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 97–102, 2017.

[73] J. Schlechtriemen, A. Wedel, J. Hillenbrand, G. Breuel, and K. Kuhnert. A lane change detection approach using feature ranking with maximized predictive power. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 108–114, 2014.

[74] S. A. Zekany, R. G. Dreslinski, and T. F. Wenisch. Classifying ego-vehicle road maneuvers from dashcam video. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1204–1210, 2019.

[75] Halil Beglerovic, Thomas Schloemicher, Steffen Metzner, and Martin Horn. Deep learning applied to scenario classification for lane-keep-assist systems. *Applied Sciences*, 8:1, 12 2018.

[76] Kai Liang, Jun Wang, and Abhir Bhalerao. Lane change classification and prediction with action recognition networks, 2022.

[77] T. Hülnhagen, I. Dengler, A. Tamke, T. Dang, and G. Breuel. Maneuver recognition using probabilistic finite-state machines and fuzzy logic. In *2010 IEEE Intelligent Vehicles Symposium*, pages 65–70, 2010.

[78] He Zhang, Zhixiong Nan, Tao Yang, Yifan Liu, and Nanning Zheng. A driving behavior recognition model with Bi-LSTM and Multi-Scale CNN. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 284–289, 2020.

[79] CarMaker | IPG Automotive — ipg-automotive.com. https://ipg-automotive.com/en/products-solutions/software/carmaker/, 2023. [Accessed 05-Jul-2023].

[80] CARLA Team. CARLA — carla.org. https://carla.org/, 2023. [Accessed 05-Jul-2023].

[81] W. Nelson. Continuous-curvature paths for autonomous vehicles. In *Proceedings, 1989 International Conference on Robotics and Automation*, pages 1260–1264 vol.3, 1989.

[82] Bin Zhou, Yunpeng Wang, Guizhen Yu, and Xinkai Wu. A lane-change trajectory model from drivers' vision view. *Transportation Research Part C: Emerging Technologies*, 85:609–627, 12 2017.

[83] Vjekoslav Diklić and Ivan Marković. Lane change classification with neural networks for automated conversion of logical scenarios. In Ivan Petrovic, Emanuele Menegatti, and Ivan Marković, editors, *Intelligent Autonomous Systems 17*, pages 778–789, Cham, 2023. Springer Nature Switzerland.

[84] Lukas Hartjen, Robin Philipp, Fabian Schuldt, Falk Howar, and Bernhard Friedrich. Classification of driving maneuvers in urban traffic for parametrization of test scenarios. 11 2019.

[85] Hanwool Woo, Yonghoon Ji, Hitoshi Kono, Yusuke Tamura, Atsushi Yamashita, and Hajime Asama. Detection method of lane change intentions in other drivers using hidden markov models. *The Abstracts of the international conference on advanced mechatronics : toward evolutionary fusion of IT and mechatronics : ICAM*, 2015.6:253–254, 12 2015.

[86] Yu-Chen Lin, Ha-Ly Thi Nguyen, and Cheng-Hsien Wang. Adaptive neuro-fuzzy predictive control for design of adaptive cruise control system. In *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)*, pages 767–772, 2017.

[87] H. Ashfaq, S. Maqbool, M. A. Khan, M. Rahman, I. Mehmood, and H. Song. Dual attention Bi-LSTM based approach for lane change prediction in autonomous vehicles. *Applied Sciences*, 11(17):7930, 2021.

[88] H. Zhou, H. Zhu, J. Chen, D. Xie, and F. Wang. Lane change intention recognition of intelligent vehicles based on multi-task learning. *Applied Sciences*, 13(18):10157, 2023.

[89] Y. Liu, Y. Zhang, K. Li, J. Wang, and Y. Xu. Lane change intention prediction of surrounding vehicles using deep learning models. *Sensors*, 22(18):6997, 2022.

[90] Sam Phillip et al. Vehicle motion classification using neural networks. *IEEE Transactions on Intelligent Transportation Systems*, 18(7):1835–1845, 2017.

[91] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[92] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Li Wang, Gang Wang, Jianfei Cai, and Tsuhan Chen. Recent advances in convolutional neural networks, 2017.

[93] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description, 2016.

[94] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[95] Kyongmin Yeo. Short note on the behavior of recurrent neural network for noisy dynamical system. *CoRR*, abs/1904.05158, 2019.

[96] dSPACE GmbH. *dSPACE ASM Scenario Editor*. dSPACE GmbH, 2023. [Accessed on June 17, 2023].

CURRICULUM VITAE

Vjekoslav Diklić was born in Đakovo, Republic of Croatia, in 1993. He completed his high school education at the Elektrotehnička i prometna škola Osijek in 2012, specializing as an electrical engineering technician. During his high school education, he participated in young innovators' competitions, including the 35th INOVA Osijek exhibition and the 3rd National Exhibition of Young Innovators and Technical Creativity organized by the Community of Technical Culture Rijeka. He obtained the Bachelor's degree in Electrical Engineering and Information Technology in 2016 and the Master's degree in Electrical Engineering and Information Technology in 2018 from the Faculty of Electrical Engineering and Computing, University of Zagreb (UNIZG-FER). During his graduate studies, he participated in the MED 18 Conference Industrial Challenge, where his team achieved second place in the Automotive Challenge. After completing his graduate studies, he joined dSPACE, where he began his research work in late 2019 and enrolled in the doctoral study programme at the Faculty of Electrical Engineering and Computing, University of Zagreb. His research interests focus on the automotive industry and simulation-based testing, with particular emphasis on vehicle trajectory analysis and classification, as well as trajectory modelling and filtering for simulation-based testing. During his work at dSPACE, he contributed to the development of industrial solutions for simulation-based testing of driver assistance systems and automated driving, and is a co-author of two patent applications in the field of automatic generation and classification of logical traffic scenarios. He is the author and co-author of several scientific and technical papers published in the proceedings of international conferences, as well as one paper published in an international scientific journal.

LIST OF PUBLICATIONS

6.0.1 *Journal Publications:*

1. Lateral classification of vehicle behavior for automated logic scenario generation. Published in: Automatika Journal for Control, Measurement, Electronics, Computing and Communications,*

Conference Publications:

1. Vehicle trajectory filtering based on optimal control and simulator-based reconstruction. Published in: 2025 48th MIPRO ICT and Electronics Convention (MIPRO),*
2. Comparison of Linear and Nonlinear Model Predictive Control for Vehicle Path Following. Published in: 2024 47th MIPRO ICT and Electronics Convention (MIPRO)
3. Effects of Applying Identified Road Lane Lines on Vehicle Autopilot Model Driving Performance. Published in: 2023 46th MIPRO ICT and Electronics Convention (MIPRO)
4. Lane Change Classification with Neural Networks for Automated Conversion of Logical Scenarios. Published in: 2022 17th International Conference on Intelligent Autonomous Systems (IAS-17),*

Publications marked with * belong to this thesis research field.

ŽIVOTOPIS

Vjekoslav Diklić rođen je u Đakovu, Republika Hrvatska, 1993. godine. Elektrotehničku i prometnu školu Osijek završio je 2012. godine, smjer elektrotehničar. Tijekom srednjoškolskog obrazovanja sudjelovao je na natjecanjima mladih inovatora, uključujući 35. INOVA Osijek te 3. Nacionalnu izložbu mladih inovatora i tehničkog stvaralaštva Zajednice tehničke kulture Rijeka. Zvanje prvostupnika elektrotehnike i informacijske tehnologije te zvanje magistra elektrotehnike i informacijske tehnologije stekao je 2016. i 2018. godine na Sveučilištu u Zagrebu, Fakultetu elektrotehnike i računarstva (UNIZG-FER). Tijekom diplomskog studija 2018. godine sudjelovao je na industrijskom izazovu u sklopu konferencije MED 18, gdje je njegov tim osvojio drugo mjesto. Nakon završetka diplomskog studija zapošljava se u tvrtki dSPACE, gdje krajem 2019. godine započinje istraživački rad te upisuje doktorski studij na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu. Njegovi istraživački interesi usmjereni su na područje automobilske industrije i simulacijskog testiranja, s posebnim naglaskom na analizu i klasifikaciju putanja vozila te modeliranje i filtriranje trajektorija za simulacijsko testiranje. Tijekom rada u tvrtki dSPACE sudjelovao je u razvoju industrijskih rješenja za simulacijsko testiranje sustava potpore vozaču i automatizirane vožnje, pri čemu je koautor dvaju patentnih prijava u području automatske generacije i klasifikacije logičkih prometnih scenarija. Autor je i koautor nekoliko znanstvenih i stručnih radova objavljenih u zbornicima radova međunarodnih konferencija, kao i jednog rada objavljenog u međunarodnom znanstvenom časopisu.