УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Никола Сакић

Примена фузије сензора у детекцији објеката и процени раздаљине у систему за избегавање чеоног судара

МАСТЕР РАД

Нови Сад, 2020.

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:			
Идентификациони број, **ИБР**:			
Тип документације, **ТД**:		Монографска документација	
Тип записа, **ТЗ**:		Текстуални штампани материјал	
Врста рада, **ВР**:		Дипломски – мастер рад	
Аутор, **АУ**:		Никола Сакић	
Ментор, **МН**:		Доц. др Момчило Крунић	
Наслов рада, **НР**:		Примена фузије сензора у детекцији објеката и процени раздаљине у систему за избегавање чеоног судара	
Језик публикације, **ЈП**:		Српски / ћирилица	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		2020	
Издавач, **ИЗ**:		Ауторски репринт	
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6	
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		5/48/16/1/34/1/0	
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кључне речи, **ПО**:		Аутомобилска индустрија; Аутономна божња; Детекција објекта; Процена раздаљине; Систем за избегавање чеоног судара	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		Један од главних изазова са којим се сусреће аутомобилска индустрија представља аутономна вожња. Развијају се бројни системи за помоћ возачу приликом вожње (енг. *ADAS*). Један од таквих система представља и систем за избегавање чеоног судара. У циљу имплементације једног оваквог система, у раду је представљено једно решење за детекцију објекта које се налази непосредно испред возила и процењује његову удаљеност. Решење је засновано на употреби фузије података са камере и *LIDAR* сензора. Платформа коришћена током израде овог решења је *Autoware*, која је базирана на *ROS* оперативном систему. Валидација решења је реализована помоћу Carla симулатора.	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	Проф. др Небојша Пјевалица	
	Члан:	Доц. др Немања Недић	Потпис ментора
	Члан, ментор:	Доц. др Момчило Крунић	

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Master Thesis	
Author, **AU**:		Nikola Sakić	
Mentor, **MN**:		Momčilo Krunić, PhD	
Title, **TI**:		Sensor fusion application in object detection and distance estimation in forward collision avoidance system	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Republic of Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:		2020	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		5/48/16/1/34/1/0	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		Automotive; Autonomous driving; Object detection; Distance estimation; Forward Collision Avoidance System	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		One of the main challenges facing the automotive industry is autonomous driving. Numerous of ADAS systems are being developed these days. One such system is the forward collision avoidance system. In order to implement such a system, this paper presents a solution for detecting an object located directly in front of the vehicle and estimating its distance. The solution is based on the usage of camera and LIDAR sensor fusion. The platform used during the development of this solution is Autoware, which is based on the ROS operating system.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	Prof. Nebojša Pjevalica, PhD	
	Member:	Asst. Prof. Nemanja Nedić, PhD	Menthor's sign
	Member, Mentor:	Asst. Prof. Momčilo Krunić, PhD	

Захвалност

Захваљујем се пре свега мојој породици и пријатељима на пруженој подршци током комплетног студирања.

Велика захвалност иде како ментору др. Момчилу Крунићу, тако и техничким менторима Стевану Стевићу и Марку Драгојевићу на стручној помоћи током израде овог мастер рада.

Између осталог захваљујем се и институту “РТ-РК”, на пруженим условима током похађања комплетних мастер студија, као и током израде самог мастер рада.

САДРЖАЈ

Списак слика

Списак табела

Скраћенице

ADAS - *Advanced Driver Assistance Systems,* напредни системи за помоћ возачу

LIDAR - *Light Identification Detection and Ranging*, ласерски сензор

AI - *Artificial Intelligence*, вештачка интелигенција

CAS - *Collision Avoidance System*, систем за спречавање колизије

ROS - *Robot Operating System,* робот оперативни систем

SAE - *Society of Automotive Engineers,* друштво инжењера

ACC - *Adaptive Cruise Control,* адаптивни темпомат

OEM - *Original Equipment Manufacturer*, власник коначног производа

ISO - *International Standardisation Organisation,* интернационална организација за стандардизацију

ASIL - *Automotive Safety Integrity Level,* систем за класификацију ризика

AUTOSAR - *Automotive Open System Architecture,* стандардизована софтверска архитектура

RTE - *Runtime Environment,* софтверски слој апстракције

VFB - *Virtual Functional Bus*, виртуална магистрала

BSW - *Basic Software*, базни софтверски слој

ECU - *Electronic Control Unit*, електронска контролна јединица

MCAL - *Microcontroller Abstraction Layer,* слој за апстракцију микро-контролера

CDD - *Complex Device Driver*, слој задужен за контролу периферних уређаја

IoT - *Internet of Things,* интернет ствари

ARA - *AUTOSAR Runtime for Adaptive Applications*, софтверски слој апстракције за адаптивну платформу

APF - *Adaptive Platform Foundation*, скуп модула који обезбеђује основне функционалности адаптивне платформе

APS - *Adaptive Platform Services*, стандардни скуп сервиса адаптивне платформе

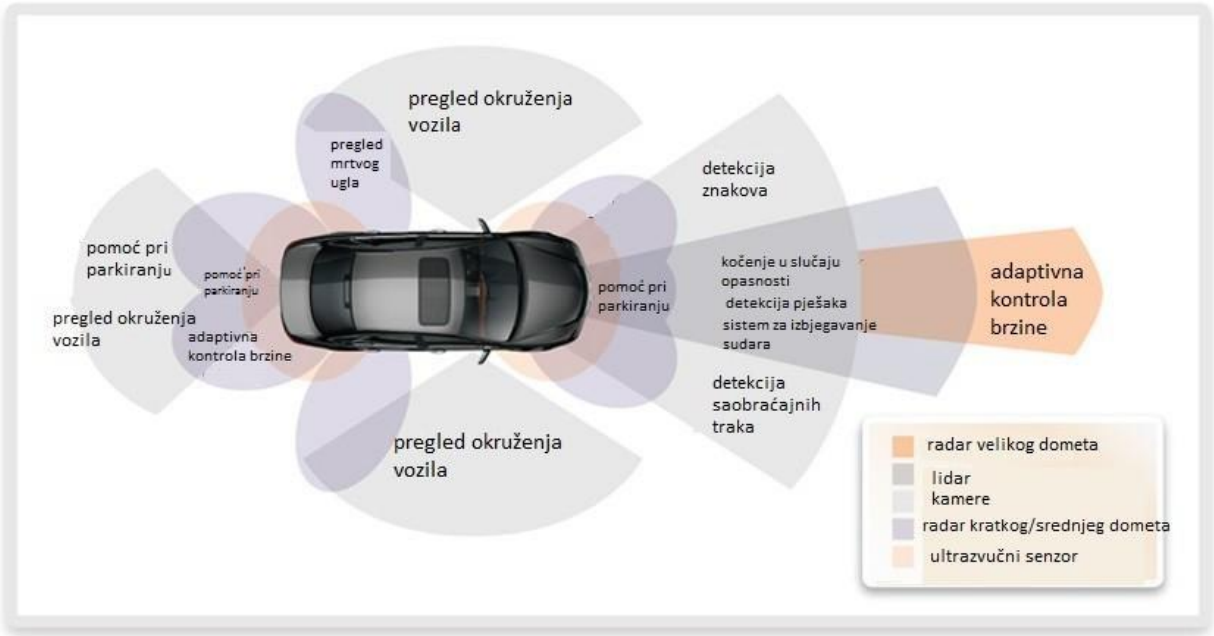
API - *Application Programming Interface*, апликативна програмска спрега

RANSAC - *Random Sample Consensus*, алгоритам за детекцију линија

YOLO - *You Only Look Once*, алгоритам дубоког учења за детекцију објеката на слици

1. Увод

Све већи број компанија покушава да на тржиште доведе решење за скроз аутономна возила, која би у потпуности одменила класичног возача. Чињеница је да и поред бројних предвиђања и дан данас не постоји право решење за потпуно аутономно возило. Још није јасно када ће заиста будућност без класичних возача доћи. Иако би се могло рећи да би употреба аутономних возила била безбеднија него случај кад возилом управља возач, велико је питање да ли би била безбеднија и од ситуације где возач има помоћ у вожњи од стране напредних система за помоћ возачу (енг. *Advanced Driver Assistance Systems – ADAS*).



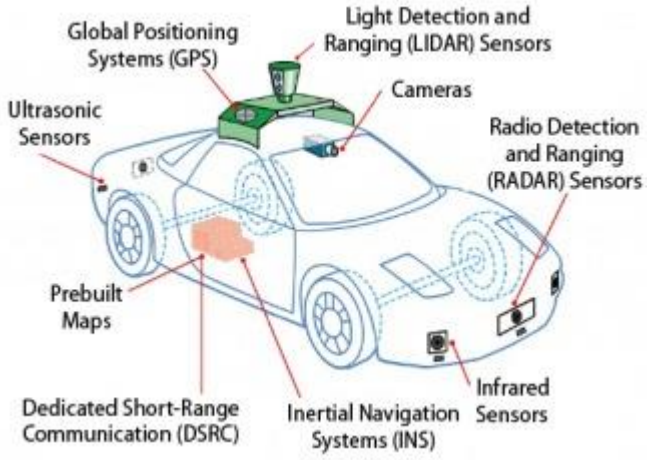
Слика 1.1 Приказ неких од типичних *ADAS* система

Овакви системи су у одређеној мери постали део наше свакодневице. Аутомобили који се данас производе садрже бар неку од приказаних функционалности (Слика 1.1), у зависности од пакета опреме. Међутим ови системи се и даље већински ослањају на присуство возача, опис нивоа аутономне вожње биће дат у поглављу 2.1.

И даље постоје неки фундаментални изазови који онемогућавају безбедно увођење комплетно аутономне технологије, која не захтева никакву управљачку интеракцију са возачем. Пет највећих преосталих препрека је наведено у наставку:

1. Сензори

   Аутономна возила користе широк спектар доступних сензора, како би вршили перцепцију околине самог возила. У то спада детекција објеката као што су пешаци, друга возила и саобраћајни знакови. Камера омогућава детекцију објеката, *LIDAR* користи ласере како би измерио дистанцу између објекта и самог возила, док је радар у могућности да, поред детекције самог објекта, прати његову брзину и правац.



Слика 1.2 Распоред сензора на аутомобилу

Ови сензори похрањују све те податке у контролни систем самог возила, који потом на основу тих података доноси одлуку о томе где возило треба да иде или када да закочи. Потпуно аутономна возила захтевају скуп сензора који прецизно детектује објекте, дистанцу, брзину ..., при свим временским условима и окружењима, без потребе да човек реагује.

Лоше време, густ саобраћај, саобраћајни знакови ишарани графитима, све то може да има негативан утицај на прецизност саме детекције. Радар који Тесла користи мање је подложан лошим временским условима, али и даље постоје изазови у гаранцији да одабрани скуп сензора, који се користе у потпуно аутономном возилу, може детектовати све објекте са потребним нивоом сигурности. Да би

постојала могућност увођења потпуно аутономних возила, где је човек ослобођен било какве управљачке интеракције са возилом, ови сензори морају бити способни да успешно раде у свим временским условима, на било којој локацији.

2. Машинско учење

   Већина аутономних возила ће користити вештачку интелигенцију и машинско учење за обраду података са претходно наведених сензора, као и за доношење одлука о наредним акцијама које треба да буду предузете. Ти алгоритми ће помагати у детекцији објеката и њиховој класификацији на основу спроведених тренинга. Систем ће потом на основу тих информација одлучивати да ли треба да предузме одређену акцију, као што је заустављање или скретање у циљу избегавања колизије са другим учесницима у саобраћају.

Слика 1.3 *AI*-а у аутомобилској индустрији

Требало би да у будућности машине буду способне да обављају детекцију и класификацију много прецизније него човек. Тренутно не постоји широко прихваћена основа која би гарантовала да су алгоритми машинског учења, коришћени у аутомобилима, сигурни. И дан данас не постоји званичан договор у оквиру индустрије или неког стандардизационог тела о томе како ови алгоритми треба да буду тренирани, тестирани или верификовани.

3. Отворен пут

   Једном кад аутономно возила буде пуштено у саобраћај, оно наставља да учи. То возило ће се кретати новим путевима, у сценаријима која нису била покривена током тренинга, те ће такође бити предмет ажурирања новог софтвера.

   Питање је како можемо гарантовати да ће систем наставити да буде подједнако сигуран, као и његова претходна верзија. Неопходно је доказати да ће свако ново учење нашег система бити у најмању руку подједнако сигурно и да систем неће

заборавити неко од претходно научених безбедних деловања. То је нешто о чему индустрија још није постигла договор.

4. Регулација

   Ефикасни стандарди и регулације које покривају један комплетан аутоматизован систем, не постоје. Тренутни стандарди који се тичу безбедности самих возила, подразумевају присуство возача, како би преузео контролу над возилом у критичним ситуацијама.

   Што се тиче возила без возача, постоје прописи за одређене функције као што су аутоматизовани системе за одржавање возила у возној траци. Постоји и међународни стандард за аутономне системе који укључује аутономна возила, који поставља релевантне захтеве, али не решава проблеме сензора и машинског учења који су наведени горе.

   Без признатих прописа и стандарда, ниједан аутомобил који се вози самостално, без обзира да ли се сматра сигурним или не, неће изаћи на отворен пут.

5. Социјална прихватљивост

   Било је бројних великих несрећа у којима су учествовали бројна аутоматизована и аутономна возила. Социјална прихватљивост није само проблем за оне који би користили таква возила, већ такође и за оне који деле цесту са њима. Јавност треба да буде укључена у одлуке о увођењу и усвајању аутономних возила. Без тога ризикујемо одбацивање ове технологије.

   Прва три од ових изазова морају бити решена, како би били у могућности да превазиђемо наредна два. Такође постоји конкуренција на тржишту у погледу тога ко ће први да уведе потпуно аутономно возило, али без сарадње у погледу тога како начинити аутомобил безбедним, пружити доказе о тој безбедности и сарадњи са регулаторним телима и јавношћу, како би добили одобрење, ови аутомобили ће остати на тест стази још дужи период.

У оквиру овог поглавља представљени су неки од главних изазова и разлога зашто на путевима широм света немамо још аутомобиле који не захтевају присуство возача. Међутим већ неко време на тржишту постоје системи који помажу возачу у управљању самим возилом. Такви системи се називају *ADAS* системи. Ти системи су у могућности да сигнализирају возачу да постоји могућност појаве неке опасне ситуације или да у одређеним ситуацијама компјутерски систем сам реагује и безбедно заустави возило.

Идеја овог рада јесте да представи једно решење имплементације система за избегавање судара (енг. *Collision Avoidance System - CAS*), базираног на употреби фузије

сензора. Овај систем представља један од значајних изазова у аутомобилској индустрији, а задужен је да у критичним ситуацијама, када постоји могућност колизије возила са неком препреком која се налази непосредно испред возила, адекватно реагује и безбедно заустави возило. Решење које ће бити објашњено у овом раду, базира се на употреби камере и *LIDAR*-а, као примарних типова сензора. Овај модул ће бити реализован у форми *ROS* пакета са применом *ROS* базиране *AUTOWARE* платформе. У сврху тестирања је кориштено симулирано окружење (*CARLA* симулатор). Рад се састоји из следећих целина.

1. Теоријске основе – Представљена основна архитектура једног аутомобилског софтвера, као и сам процес његовог развоја. Такође су објашњени сви теоријски концепти коришћени током израде овог решења, као и само окружење које је кориштено у ову сврху.
2. Решење – У оквиру овог поглавља је дат опис целог система који је креиран у сарадњи са осталим члановима тима, те како је *CAS* систем спрегнут са остатком система. Дат је детаљан опис архитектуре и како су концепти описани у поглављу теоријске основе, спрегнути у заједничку целину.
3. Поступак испитивања и резултати – Опис самог окружења коришћеног у сврху верификације добијеног решења, као и детаљан увид у сценарије под којим је ово решење тестирано.
4. Закључак – За крај је дат још један преглед свега тога шта је постигнуто у оквиру овог рада, као и шта је то добро били, а шта је то што би могло да представља проблем. Такође је представљен и план за даље усавршавање овог решења.

2. Теоријске основе

У оквиру овог поглавља ће бити описани основни концепти коришћени током израде овог рада. За почетак ће бити дат опис саме програмске подршке са применом у аутомобилској индустрији, као и нивои аутоматизације. Након тога ће акценат бити стављен на саме алгоритме који су коришћени за обраду података са камере и *LIDAR*-а, као и теоријски концепт који стоји иза њихове фузије. Као незаобилазна целина приликом пројектовања једног софтвера за паметне аутомобиле, јесте само окружење које је коришћено за сам развој. До недавно су у развој аутомобилског софтвера били укључени само произвођачи аутомобила, као и њихови добављачи, међутим развој оваквог софтвера није био приступачан за академска истраживања и ентузијастима. Захваљујући *Autoware* платформи отвореног кода (енг. *Open source*), омогућено је да процес развоја програмске подршке за паметне аутомобиле буде приступачнији него икад. У сврху тестирања коришћено је симулирано окружење (*CARLA*), услед недостатка реалних услова за спровођење тестова.

2.1 Нивои аутономне вожње

Систем класификације [1] је постављен од стране *SAE International* (енг. *Society of Automotive Engineers*) заједнице [2] 2014. године и често се користи као референтна тачка у свим дискусијама по питању аутоматизације возила.

Ниво 0 – Односи се на и даље уобичајена возила, где возач предузима све операције, како би управљао возилом. Ту спадају: перцепција околине, управљање, убрзавање и кочење, без било какве помоћи од стране неког аутоматизованог рачунарског система.

Ниво 1 – На овом нивоу возач и даље рукује већином функционалности, али уз неку минималну помоћ самог возила. Оваква возила већ поседују *ACC* систем за контролу брзине

и растојања. Поред овог система, ови аутомобили могу поседовати и систем за помоћ при паркирању, где аутомобил даје звучни сигнал возачу у ситуацији да се возило налази близу неке препреке.

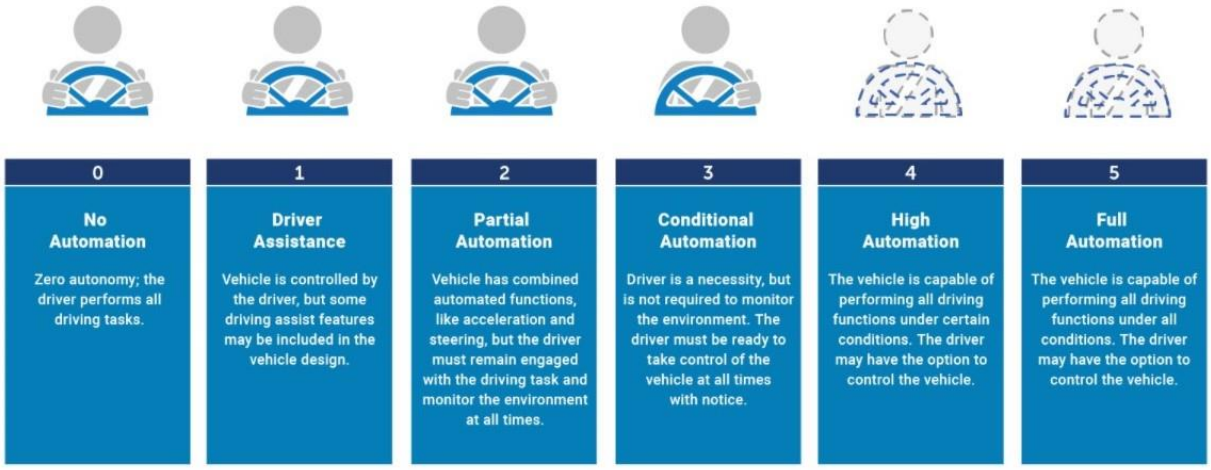
Ниво 2 – Парцијална аутоматизација, која омогућује да возач буде ослобођен од обављања неких функција, као што су управљање и контрола брзине, међутим возач мора у свакој ситуацији да буде у приправности и држи обе руке на волану, како би преузео контролу уколико је то неопходно.

Овај рад имплементира функционалности управо овог нивоа, а то су центрирање возила у возној траци, док систем за контролу брзине води рачуна о одржавању безбедне дистанце од других возила. Суштинска разлика између нивоа 1 и 2 је у томе да на нивоу 2 наведена два ADAS система могу да раде у исто време, што није случај на нивоу 1.

Ниво 3 – На овом нивоу возило може самостално извршавати све функционалности управљања, под одређеним околностима. У тим околностима, возач мора бити спреман да преузме контролу над возилом, у одређеном временском оквиру, када систем за аутономну вожњу то од њега захтева. У свим осталим околностима возач и даље директно врши контролу над возилом.

Ниво 4 – Систем за аутономну вожњу може самостално обављати све задатке и надгледати околину возила, али и даље под одређеним околностима. Када су ти услови испуњени, возач не мора обраћати пажњу на пут.

Ниво 5 – Систем за аутономну вожњу, може потпуно самостално управљати возилом у свим условима. Људи су у овом случају само путници, које ће аутомобил самостално превести из тачке А у тачку Б. Било каква директна управљачка спрега човека са возилом више није неопходна (волан, педале гаса и кочнице …). Возач само наводи адресу крајњег одредишта.



Слика 2.1 Нивои аутономне вожње

2.2 Програмска подршка са применом у аутомобилској индустрији

Данашњи механички оријентисани аутомобили се брзо трансформишу у софтверски дефинисане транспортне платформе. Најновије аутомобилске иновације се баве увођењем интуитивних информативно забавних (енг. *Infotainment*) система, могућност вожње без присуства возача и електрификацијом. Будућа возила ће све мање зависти од механичких компоненти, већ ће све бити базирано на програмској подршци. Ова се промена тако брзо дешава да су *OEM* и Tier1 компаније под сталним притиском како би одржале корак са индустријом.

Велики број компанија из аутомобилске индустрије, преузима софтверске модуле који су већ нашли примену и у другим индустријама. У тај софтвер спадају како сами оперативни системи, тако и већина софтвера средњег нивоа (енг. *Middleware*). На тај начин значајно смањују временске рокове и трошкове развоја, јер је креирање оригиналног кода много теже. Многе технологије коришћене у сврху информативно забавних система потичу из индустрије паметних телефона. *ADAS* системи су првобитно позајмљени из програмске подршке намењене за авио и производну индустрију. Данас су изразито постали популарни оперативни системи за рад у реалном времену који су преузети од компанија које се баве производњом полупроводничких компоненти и компанија које се баве уграђеним системима (енг. *Embedded Systems*).

Општи развојни изазови постају врло реални. Како се софтверски садржај повећава у готово сваком делу возила, тако се чини и напор који је потребан да би различити системи функционисали као кохезивна целина. Играчи који се такмиче у аутомобилској индустрији имају изразито тежак задатак данас. На пример, модерним системима за забаву сада је потребно више од три године за развој, при чему неколико стотина софтверских инжењера доприноси свакој итерацији. Од целог уложеног труда од 30% до 50% чини интеграција. Имајући у виду широку мрежу добављача који су укључени у развој. За промене било ког софтверског модула често је потребна велика обрада. Како би ови системи остали савремени, захтева се обнова сваких неколико година, како би остали у складу са новим функционалностима и перформансама.

2.2.1 Захтеви у погледу квалитета и сигурности

Строги стандарди квалитета аутомобила узимају у обзир да је животни циклус возила знатно дужи од животног века једног паметног телефона. Штавише, системи у оквиру возила морају бити поуздани у много изазовнијим условима (нпр. изложеност временским условима), из тог разлога примењују се строги функционални безбедносни захтеви, како је

то наведено у *ISO* стандарду 26262 [3]. Велики број безбедносно критичних функција, које могу довести до угрожавања безбедности учесника у саобраћају, мора бити пројектовано са посебном пажњом.

ASIL класификације ризика диктирају сигурносне захтеве софтвера, којих се *OEM*-ови и *Tier1* добављачи морају придржавати. Трошкови интеграције, тестирања, провере и верификација функција за испуњавање ових захтева могу лако да достигну 40% или више укупног развојног буџета.

2.2.2 Колаборација између различитих *OEM*-ова и *Tier1* добављача

Данас, од различитих *OEM* компанија, *Tier1* добављача, произвођача полупроводничких компоненти доносе на тржиште различите сервисе и производе који се баве развојем софтвера за аутомобилску индустрију. Постало је поприлично фрустрирајуће да све потенцијалне муштерије чисто софтверских компанија, као што су различити *OEM*-ови и *Tier1* добављачи, користе различите стратегије приликом имплементације апликативног софтвера.

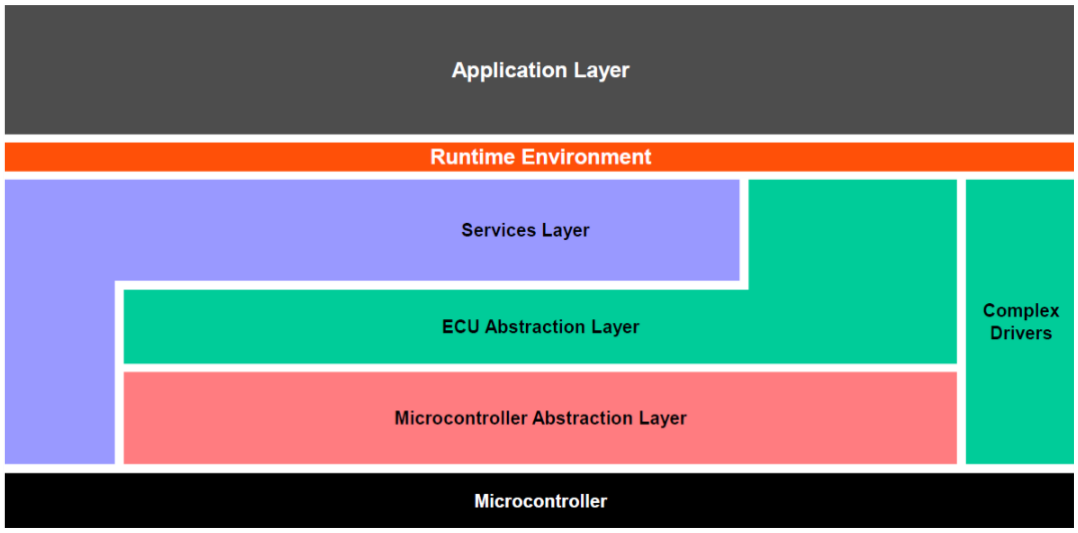
Како би се привукле компаније задужене искључиво за развој софтвера, потребна је већа стабилност и усаглашеност заинтересованих страна у аутомобилској индустрији. У ту сврху је од пресудног значаја стандардизација платформе за лакшу интеграцију софтвера, као и успостављање модела сарадње и технолошких веза унутар индустрије, како би се *OEM* произвођачи, добављачи и софтверске компаније усагласиле. Досадашњи напори у овом правцу укључивали су платформе *AUTOSAR Classic* и AUTOSAR *Adaptive,* а воде их углавном европски *OEM*-ови и *Tier1* добављачи.

Они имају за циљ да успоставе стандарде сарадње у целој индустрији и учврсте јасне улоге за развој софтвера у возилу, што је у интересу свих страна. Исто се односи на постизање стандардизованих хардверско-софтверских спрега (енг. *Interfaces*) и спрега унутар софтверске гомиле (енг. *Stack*), који ће заузврат помоћи у смањењу трошкова верификације олакшавањем поновне употребе софтвера.

2.2.3 AUTOSAR Classic

Главна улога стандардизоване *AUTOSAR Classic* [4] софтверске архитектуре (Слика 2.2) је раздвајање хардверски независног апликативног софтвера (енг. *Application Layer*) и хардверски оријентисаног базног софтвера (енг. *Basic Software - BSW*) од стране слоја за апстракцију (енг. *Runtime Environment - RTE*). Са горње стране *RTE* слоја, овај слој апстракције омогућава развој *OEM* специфичних апликација, док на доњој страни *RTE*-а омогућава стандардизацију и развој *OEM* независног основног софтвера. Једна од главних

карактеристика *AUTOSAR* софтверске архитектуре јесте скалабилност *ECU* софтвера, која омогућава дистрибуцију функционалних софтверских модула између различитих *ECU*-ова, као и могућност интеграције софтверских модула из различитих извора.



Слика 2.2 *AUTOSAR Classic* архитектура

Слој апстракције као софтвер средњег нивоа врши интеграцију појединачних апликација са базним софтвером. Он је задужен да обезбеди комуникацију и размену података између различитих функција (енг. *Runnables*). Овај слој је неопходан како би се обезбедио развој хардверски независних апликација (енг. *Software Components - SWC*). Слој апстракције своју функционалност заснива на употреби виртуелне функционалне магистрале (енг. *Virtual Functional Bus - VFB*), која раздваја апликације од саме инфраструктуре. За апликацију није од значаја како нижи слојеви рукују подацима, једино што је од значаја за апликацију јесте, који подаци се шаљу преко којег прикључка (енг. *Port*). Са апликативне стране гледишта, није битно да ли прикључак представља везу унутар једног *ECU*-а (енг. *Intra ECU Communication*) или везу са неким другим *ECU*-ом (енг. *Inter ECU Communication*). *VFB* рукује комуникацијом како у оквиру једног *ECU*-а, тако и између више *ECU*-ова.

Испод слоја за апстракцију налази се базни софтвер, који је за разлику од апликативног слоја, зависан од саме хардверске платформе. Овај слој је стандардизован и омогућава развој *OEM* независног софтвера. *BSW* се састоји из следећа четири слоја:

1. **Слој сервиса** (енг. *Services Layer*) – Овај слој представља највиши ниво апстракције у оквиру *BSW* слоја. Овај слој обезбеђује основни скуп сервиса за апликације, *RTE* и *BSW* модуле. Сервиси као што су коришћење услуга оперативног система, комуникациони сервиси, сервиси везани за употребу меморије, као и сервиси намењени за контролу стања *ECU*-а.

2. **Слој за апстракцију *ECU*-а** (енг. *ECU Abstraction Layer*) – Апстракција у овом смислу представља прикривање детаља о самом ECU и обезбеђивању *API*-а према вишим слојевима за имплементацију функционалности. Овај слој обезбеђује приступ самим контролерима (енг. *Drivers*).

3. **Слој за апстракцију микро контролера** (енг. *Microcontroller Abstraction Layer - MCAL*) – У оквиру овог слоја се налазе сами контролери за приступ периферијама. *MCAL* је најнижи слој *BSW* софтвера и самим тим је овај слој зависан од самог микро контролера (енг. *Microcontroller Unit - MCU*). То значи да имплементација у оквиру овог слоја мења са променом *MCU*-а.

4. **Комплексни контролери** (енг. *Complex Device Drivers - CDDs*) – Овај слој обавља конекцију *SWC* компоненте из апликативног слоја директно са *MCU*-ом путем *RTE*-а. Овај слој се користи за имплементацију контролера периферних уређаја који нису дефинисани *AUTOSAR* стандардом. Овај слој је изразито зависан од хардвера и из тог разлога постоји ограничење по питању преносивости кода.

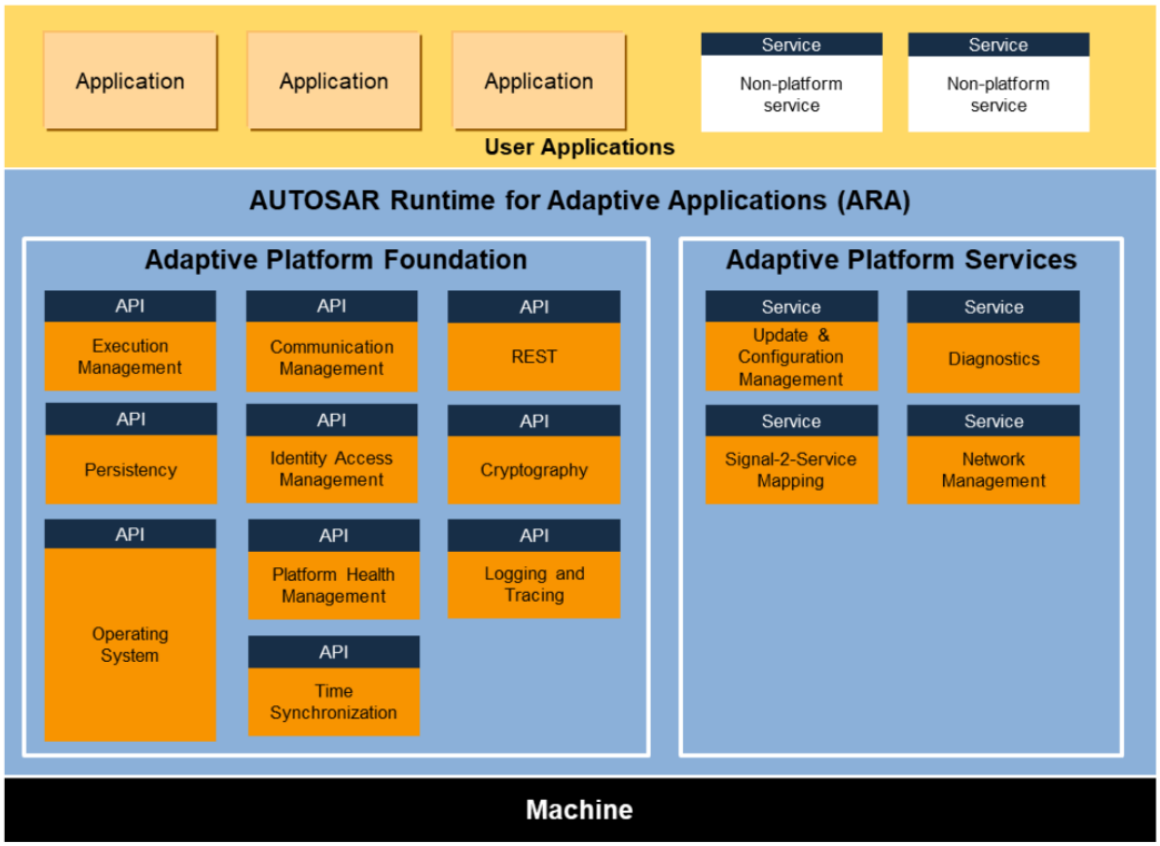
Поред софтверске архитектуре, *AUTOSAR* је увео усаглашени методолошки приступ за развој аутомобилског софтвера. Ово је углавном вођено потребом да се побољша сарадња између различитих страна укључених у данашње пројекте унутар аутомобилске индустрије. *AUTOSAR* пружа средства за специфицирање свих аспеката потребних за интеграцију софтверске компоненте на *ECU* и за интеграцију различитих *ECU*-ова у целокупну мрежну комуникацију кроз различите системе магистрала.

2.2.4 AUTOSAR Adaptive

Данашњи *ECU*-ови имплементирају функционалности које замењују електро-механичке системе у возилу. Софтвер уграђен у такве *ECU*-ове контролише неки излаз на основу примљених сигнала послатих од стране других *ECU*-ова, повезаних на мрежу унутар возила. Већина контролног софтвера је дизајнирана и имплементирана за неко конкретно возило и он се не мења током животног циклуса самог возила.

Будуће функционалности возила, као што су аутономна вожња, ће увести компјутерски захтеван софтвер у аутомобиле, који ће морати да испуни строге захтеве по питању безбедности. Такав софтвер имплементира функционалности као што су, перцепција околине, планирање кретања, и интеграција возила у једну ширу инфраструктурум као што су интернет ствари (енг. *Internet of Things - IoT*). Овакви системи захтевају могућност да софтвер унутар возилу нуди могућност ажурирања током животног циклуса, услед еволуције спољних система или побољшања функционалности.

Овакве потребе модерних *ECU*-ова не могу бити задовољене коришћењем класичне платформе. Због тога *AUTOSAR* специфицира другу софтверску платформу, тзв. адаптивну платформу (енг. *AUTOSAR Adaptive Platform*) [5]. Адаптивна платформа (Слика2.3) (АП) пружа углавном захтевна процесирања и комуникационе механизме, те нуди флексибилну конфигурацију софтвера, као што је подршка за ажурирање софтвера путем удаљеног приступа. Функционалности специфичне за класичну платформу, као што су приступ електричним сигналима и системима комуникације специфичним за аутомобиле, могу бити интегрисане у АП.



Слика 2.3 *AUTOSAR Adaptive* архитектура

Адаптивне апликације се извршавају на врху софтверске архитектуре и користе сервисе које им обезбеђује слој за апстракцију (енг. *AUTOSAR Runtime for Adaptive Applications - ARA*). *ARA* слој обезбеђује *API* за адаптивне *AUTOSAR* сервисе, као што су управљање конфигурацијом софтвера, управљање безбедношћу и дијагностика. *ARA* слој се састоји од два функционална кластера, а то су *APF* (*Adaptive Platform Foundation*) и *APS* (*Adaptive Platform Services*). *APF* функционални кластер садржи скуп основних сервиса за апликације, који апстрахују хардвер и омогућују ефективно извршавање самих апликација на датој хардверској платформи. Са друге стране *APS* кластер управља комуникацијом између *AUTOSAR* адаптивних апликација.

2.2.4.1 Будућност адаптивне платформе

Умреженост возила представља наредни велики искорак у аутомобилској индустрији. Комуникација *V2V* (*Vehicle to Vehicle*), *V2X* (*Vehicle to* Everything) [6], удаљена дијагностика и аналитика у облаку (енг. *Cloud based analytics*) су неке од актуелних тема о којима се разматра у оквиру индустрије. *V2X* системи захтевају безбедну комуникацију са другим возилима, као и са инфраструктуром, која не мора бити по *AUTOSAR* стандарду. Следећа генерација возила ће бити конектована са другим возилима, паметним телефонима, саобраћајном инфраструктуром путем *V2X* апликација, које треба да се ажурирају на даљину (енг. *Over The Air - OTA*). То је место где ће *AUTOSAR Adaptive* наћи своју примену.

Слика 2.4 *V2X* комуникација

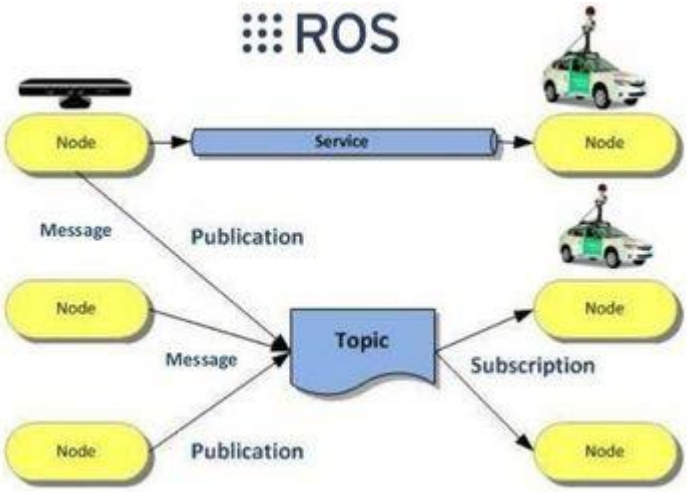
2.3 ROS

ROS [7] је робот оперативни систем отвореног кода (енг. *Open-source*). *ROS* заправо није оперативни систем у традиционалном смислу управљања процесима и компјутерским ресурсима, већ представља *Framework* који се извршава на већ постојећем оперативном систему. Он је скуп алата, библиотека и различитих конвенција за развој функционалности код разних платформи за контролу робота. *ROS* омогућава једноставан приступ подацима са сензора, обраду тих података и генерисање излаза у виду актуације. Комплетан *ROS* систем је дизајниран тако да буде лако дистрибуиран на различите рачунарске платформе.

Услед своје портабилности и једноставности, *ROS* је у великој мери заступљен и у оквиру аутомобилске индустрије, те је из тог разлога коришћено као окружење приликом

овог истраживања. Велики број кода који је развијан у оквиру *ROS* заједнице отвореног кода, за примену у роботима, директно је нашао примену и у аутомобилској индустрији. Функционалности као што су креирање *3D* мапе окружења, локализација нашег возила у простору, избегавање препрека, могу директно да буду преузете у сврху аутоматизације возилима.

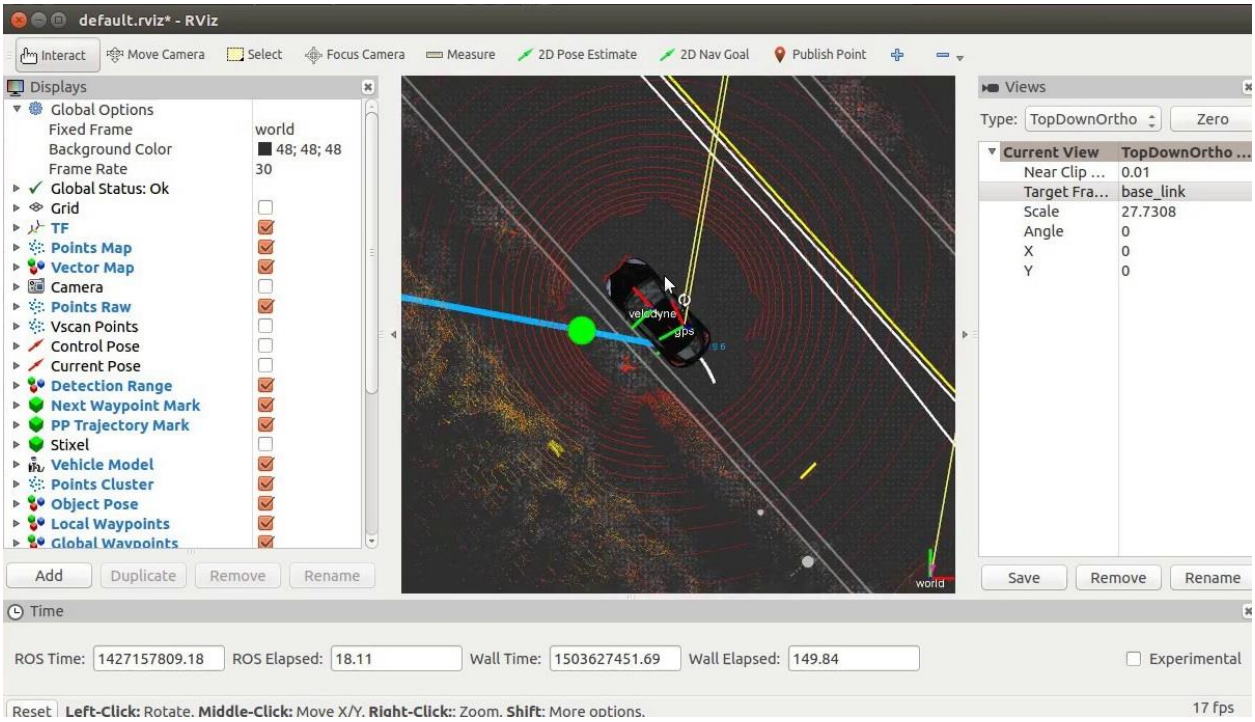
ROS је заснован на концепту чворова (енг. *Nodes*), који представљају разне процесе. Тако један чвор може да буде задужен за прикупљање података са сензора, обраду података или евентуално да буде задужен за слање команди неком актуатору. Сваки чвор приликом свог покретања се региструје код мастер чвора. Мастер је чвор чија је главна улога декларисање и регистровање осталих чворова, те се тиме омогућује осталим чворовима да пронађу једни друге и да размењују податке. У оквиру овог чвора се налази сервер за параметре (енг. Parameter server), који представља неку врсту централизоване базе података у оквиру које чворови могу складиштити своје податке. Подаци се могу размењивати на два начина између чворова (Слика 2.5).



Слика 2.5 Размена података у *ROS* окружењу

Могу размењивати асинхроно путем тема (енг. *Topics*) или синхроно путем сервиса. Тема је систем преноса података заснован на систему претплата/објављивање (енг. *Publisher/Subscriber*). Један или више чворова може објавити податке о некој теми, а један или више чворова могу читати податке о тој теми. Сама тема је типизирана, то значи да је порука која се прослеђује у оквиру неке теме, увек структуирана на исти начин. У оквиру теме се прослеђују поруке (енг. *Messages*). Порука садржи комбинацију примитивних типова (*string, bool, int, float* …). Ова метод имплементира асинхрони начин комуникације намењен за ситуације где постоји више пошиљалаца и више прималаца. Сервиси се са друге стране користе за синхрону комуникацију између два чвора.

У оквиру *ROS* окружења такође постоји формат за чување и репродукцију података, тзв. Торбе (енг. *Bags*). Овај механизам омогућава Овај механизам омогућава прикупљање података измерених сензорима и њихово репродуковање онолико пута колико је потребно да се симулирају стварни подаци. Алат који нам омогућава визуелизацију свих података у оквиру *ROS* окружења јесте *RVIZ* (*ROS Visualization*).



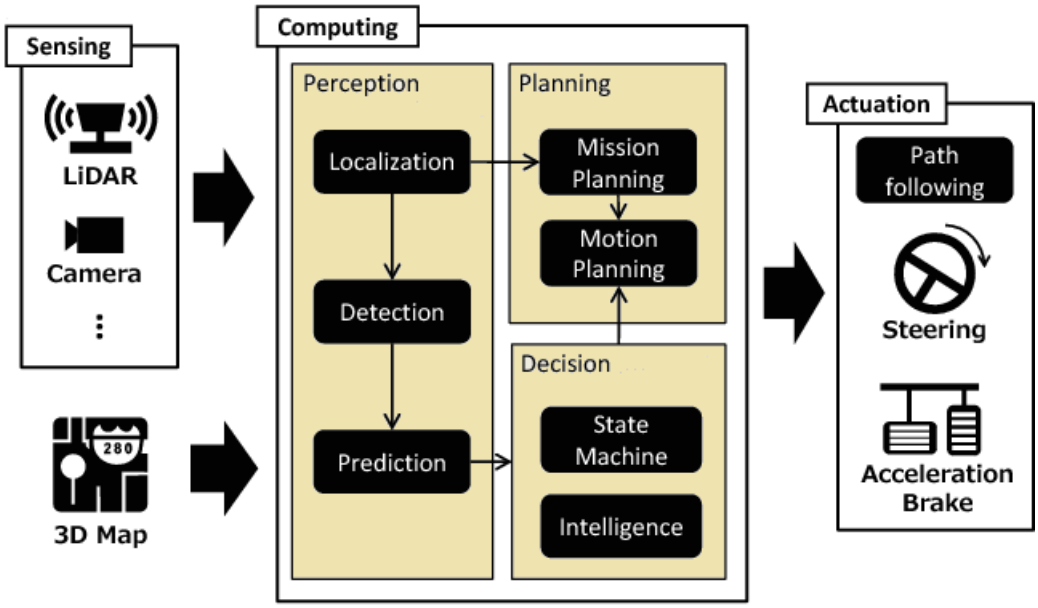
Слика 2.6 *ROS* алат за визуелизацију - *RVIZ*

2.4 Autoware

Autoware [8] представља непрофитабилну организацију, која подржава пројекте отвореног кода, намењених за развој аутономне мобилности. Циљ ове фондације, која је основана 2015. године на *Nagoya* универзитету у Јапану, јесте да студентима и другим ентузијастима, омогући развој унутар ове области. Ова платформа нуди широк спектар већ постојећих модула за аутономну вожњу, као што су локализација, детекција, предвиђање, планирање и контрола. Ова платформа заправо представља *Framework* заснован на *ROS*-у, намењен развоју и тестирању разних функционалности аутономних возила. Тренутно постоје три пројекта на којима ради ова заједница, а то су *Autoware.AI, Autoware.AUTO* и *Autoware.IO.*

Приликом истраживања које је покривено овим радом, коришћена је *Autoware.AI* платформа, која је заснована на употреби *ROS1*, а покренута је као *R&D* платформа за аутономну вожњу, са намером да се задовољи *SAE* ниво ¾. Подржава многе платформе које се налазе у возилима и садржи велики број фајлова за покретање (енг. *Launch files*),

конфигурационих фајлова, неуронски мрежа и мапа, уз вишеструку имплементацију алгоритама за контролу, локализацију, перцепцију, планирање и симулацију.



Слика 2.7 *Autoware* архитектура

Битно је да се нагласи да је *Autoware* превасходно дизајниран за урбану средину, међутим магистрални путеви и аутопутеви такође могу бити покривени, уз додатак одређених модула.

1. ***Sensing*** **–** *Autoware* углавном користи камеру и *LIDAR* за препознавање околине. Технологија на којој је *LIDAR* базиран, јесте мерење времена које је потребно да генерисани ласерски импулс дође до површине објекта и врати се назад до извора. Облак података (енг. *Point cloud*) са *LIDAR* сензора могу се искористити за генерисање дигиталних *3D* приказа скенираних објеката. Са друге стране, камере се углавном користе за препознавање скенираних објеката.
2. ***Computing*** **–** Ово представља кључну компоненту аутономних возила, која користи податке са сензора и *3D* мапе, како би возило било рачунало коначну путању и комуницирало са модулима задуженим за актуацију. Овај модул је сачињен од модула за перцепцију (енг. *Perception*), одлуку (енг. *Decision*) и планирање (енг. *Planning*).
    - ***Perception*** **–** Као што је раније речено, безбедност у аутономним возилима представља озбиљан изазов. Из тог разлога, модул за перцепцију мора прецизно да израчуна позицију его возила унутар *3D* мапе и препозна објекте у сцени, као и њихов статус.

- ***Decision*** **–** Када су препреке и саобраћајни сигнали детектовани, путање других покретних објеката могу бити процењене. Модули за планирање и одлучивање користе ове процењене резултате да би одредили одговарајући положај у који треба довести его возило. *Autoware* имплементира машину стања за разумевање, планирање и доношење одлука као одговор на тренутно стање на путу.
- ***Planning*** **–** Овај модул генерише путању на основу излаза из модула за доношење одлуке. *Autoware* одлучује глобалну путању на основу тренутне локације и одређеног одредишта

3. ***Actuation*** – Након што се одреди путања, аутономно возило мора пратити ту путању. Модул за планирање генерише команде за покретање его возила у виду *Pure-Pursuit* или *MPC* (*Model Predictive Control*) механизма.

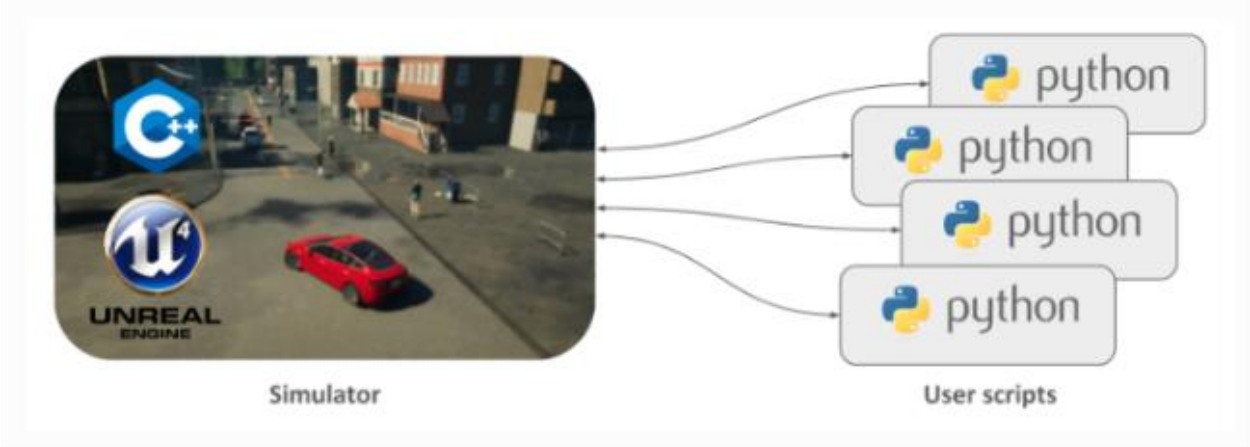
Неколико симулатора је подржано у сврху *Autoware*-а, као што су *LGSVL* и *CARLA*. У оквиру овог решења, коришћен је *CARLA* симулатор и из тог разлога ће он бити објашњен у наредном поглављу.

2.5 Carla simulator

Скуп могућих догађаја који се могу десити у току вожње је бесконачно велики, где је већина тих догађаја изразито ретка. Из тог разлога, приликом развоја аутономних возила неопходно је довести алгоритме који се развијају у велики број ситуације, које није могуће или је опасно изазвати у реалним условима. У ту сврху се користе симулирана окружења, која омогућавају креирање жељеног сценарија и тестирање развијаних алгоритама у таквом једном окружењу.

Слика 2.8 *CARLA* симулатор

Често коришћени симулатор јесте *CARLA* (*Car Learning to Act*) (Слика 2.8) [9], развијен од стране *Alexey Desovitskiy*-а из *Intel Labs*-а, неколико истраживача са *Toyota Research* института и *Computer Vision* центра у Барселони. Један од главних циљева *CARLA*-е је помоћ у истраживању и развоју аутономне вожње, која служи као алат којем корисници могу лако приступити и прилагодити. Заснована је на *Unreal Engine*-у за покретање симулације и користи *OpenDRIVE* стандард за дефинисање путева и градских подешавања. Захваљујући *ROS-bridge*-у, *CARLA* може да се интегрише у *ROS* односно *Autoware* окружење и да се различите информације из *CARLA* симулатора генеришу у форми *ROS* порука.



Слика 2.9 Задавање контрола симулатору преко *Python* скрипти

Контрола над симулацијом је дата путем *API*-а који се обрађује у *Python*-у и C++-у, који се непрестано проширује у току пројекта.

Како би ублажио процес развоја, обуке и валидације система вожње, *CARLA* је еволуирала у екосистем пројеката, који је око главне платформе изградила заједница. У том је контексту важно схватити неке ствари о томе како функционише *CARLA* како би у потпуности схватили њене могућности.

CARLA симулатор се састоји од скалабилне клијент-сервер архитектуре. Сервер је одговаран за све везано са симулацијом, приказ података са сензора, обрада физичких израчунавања, ажурирање стања свих учесника симулације и још пуно тога. Најпожељније је покретање сервера са наменским *GPU*-ом, нарочито у случају рада са алгоритмима машинског учења. Клијентска страна се састоји од скупа модула клијента који контролишу логику учесника симулације и постављање других услова саме симулације. То се постиже коришћењем *CARLA API-a* који посредује између сервера и клијента који се непрестано развија како би пружио нове функционалности. Треба напоменути неке од основних концепата *CARLA* симулатора.

1. ***Traffic manager*** – Уграђени систем који преузима контролу над возилом, осим оног који се користи за учење. Користи се за креирање урбаних средина са различитим понашањем.

2. ***Sensors*** – Возила се ослањају на њих ради дистрибуције информација о свом окружењу. Могу се по потреби додавати на его возило, те се евентуално подаци снимљени са тих сензора могу похрањивати у фајл. Тренутно пројекат подржава различите типове сензора од камера, радара и *LIDRA*-а до многих других.

3. ***Recorder*** – Ова функција се користи за поновну реакцију симулације корак по корак за сваког учесника симулације. Омогућује приступ било којем тренутку на временској траци било где у свету, чинећи га одличним алатом за праћење.

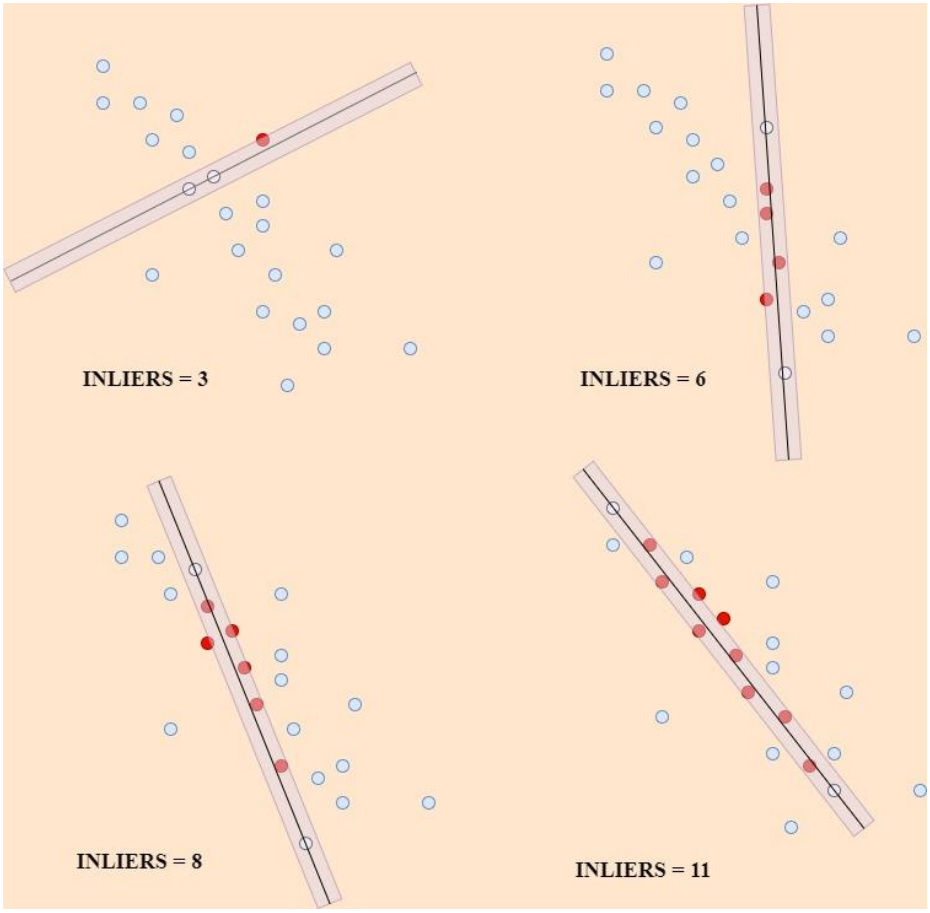
4. ***ROS bridge and Autoware implementation*** – Пројекат *CARLA* повезује чворове и ради на интеграцији симулатора у оквиру других окружења за учење.

5. ***Open assets*** – *CARLA* олакшава различите мапе за урбана подешавања са контролом временских услова и библиотеку нацрта са широким скупом актера, који се користе. Међутим, ови елементи се могу прилагодити и генерисати на основу јединствених смерница.

6. ***Scenario runner*** – Да би олакшао процес учења за возила, *CARLA* пружа низ рута које описују различите ситуације кроз које треба пролазити. Ово такође поставља основу за *CARLA challenge* отворен за све да тестирају своја решења.

2.6 RANSAC

RANSAC се користи за откривање контура, а што је још важније користи се и за детекцију линија [10]. Налази примену у обради слике и сегментацији траке, где се овај алгоритам користи за откривање линија на путу. Ово је итеративна метода за процену параметара математичког модела на основу података где постоје *outlier*-и. Са друге стране метода најмањег квадрата је осетљива на постојање *outlier*-а. Неформално, *outlier* је податак који није у складу са моделом, док се подаци који су у складу са моделом називају *inlier*-и. Из скупа доступних података се бира N података (нпр. N=2), а затим се рачуна број података који су садржани унутар задате границе добијеног правца. Овај поступак се понавља жељени број пута. Након што је достигнут специфицирани број понављања, бирамо решење са највећим бројем *inlier*-а. *RANSAC* је недетерминистички алгоритам у смислу генерисања задовољавајућег резултата са одређеном вероватноћом, која се повећава са бројем итерација алгоритма.



Слика 2.10 Итеративни поступак проналаска линије *RANSAC* алгоритмом

Овај алгоритам је веома робустан у погледу постојања *outlier*-а, где је при том једноставан и ефикасан. Потребно је дефинисати параметре као што су број понављања и величина прага за дефинисање *inlier*-а. Овај алгоритам може потрајати неко време у случају сложенијих модела и великог скупа података. У сврху обраде података са *LIDAR*-а и детекцију возне траке, неопходна је промена перспективе из које се посматра облак података (енг. *Point cloud*). Облак података се може замислити у облику *3D* паралелопипеда са одређеном границом. Како би се детектовала возна раван у форми једне линије, треба да се посматра тај паралелопипед из профила. Након тога се могу издвојити све тачке које припадају површини пута. Након што се одбаце све тачке које припадају површини пута из добијеног скупа података, поступак детекције објеката може бити настављен.

2.7 Кластеризација (*K-Means*, *Mean-shift*)

Кластеризација је техника машинског учења која укључује груписање тачака које представљају неке податке. Када постоји неки скуп података, може се искористити алгоритам кластеризације како би се свака тачка придружила одређеној групи. Теоретски, тачке које се налазе у истој групи треба да имају слична својства или карактеристике, док

би тачке у различитим групама требале да имају веома различита својства. Кластеризација [11] је метода учења без надгледања (енг. *Unsupervised learning*) и уобичајена је техника за статистичку анализу података која се користи у многим областима. У науци о подацима (енг. *Data Science*) се може користити ова анализа, како би се из скупа података одредило којој групи података припадају тачке, након што се примени алгоритам кластеризације. Овде ће бити дат опис два најзаступљенија алгоритма за кластеризацију, где ће такође бити наведене разлике између њих, на основу којих је и одабран одговарајући алгоритам.

2.7.1 *K-Means*

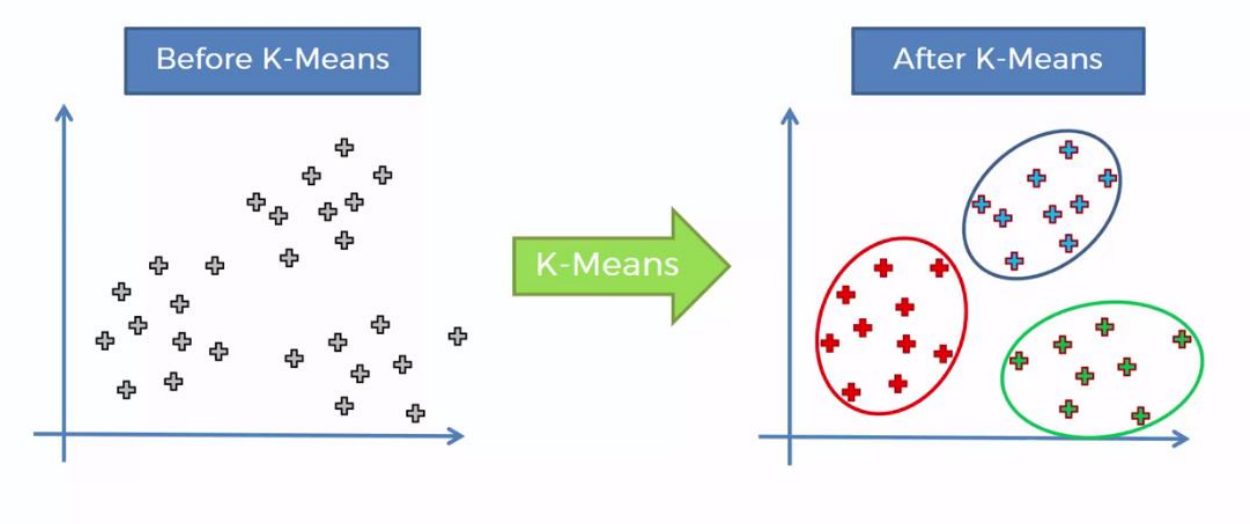
K-Means је вероватно најпознатији алгоритам за кластеризацију. Један је од основних концепата у *Data Science* и *Machine Learning* областима. Поступак је следећи.

1. Прво се селектује број класа/група, а потом се насумично иницијализују њихове централне тачке. Да би се одредио број класа, неопходно је погледати скуп података и идентификовати било какве различите групе.

2. Сваки податак се класификује мерењем удаљености те тачке и централне тачке сваке појединачне групе, након тога се тој тачки додели припадност групи чијој је централној тачки најближа.

3. На основу ових класификованих тачака, ажурирају се централне тачке сваке групе, тако што се узму средња вредност свих вектора у групи.

4. Ови кораци се понављају дефинисан број пута или до момента када се централне тачке сваке групе престану померати између неколико узастопних итерација.



Слика 2.11 *K-Mean* алгоритам кластеризације

K-Means има предност у томе што је прилично брз, јер све што се овде ради јесте израчунавање удаљености између тачака и центара сваке групе. Врло мало рачунања, стога има линеарну сложеност *O(n)*.

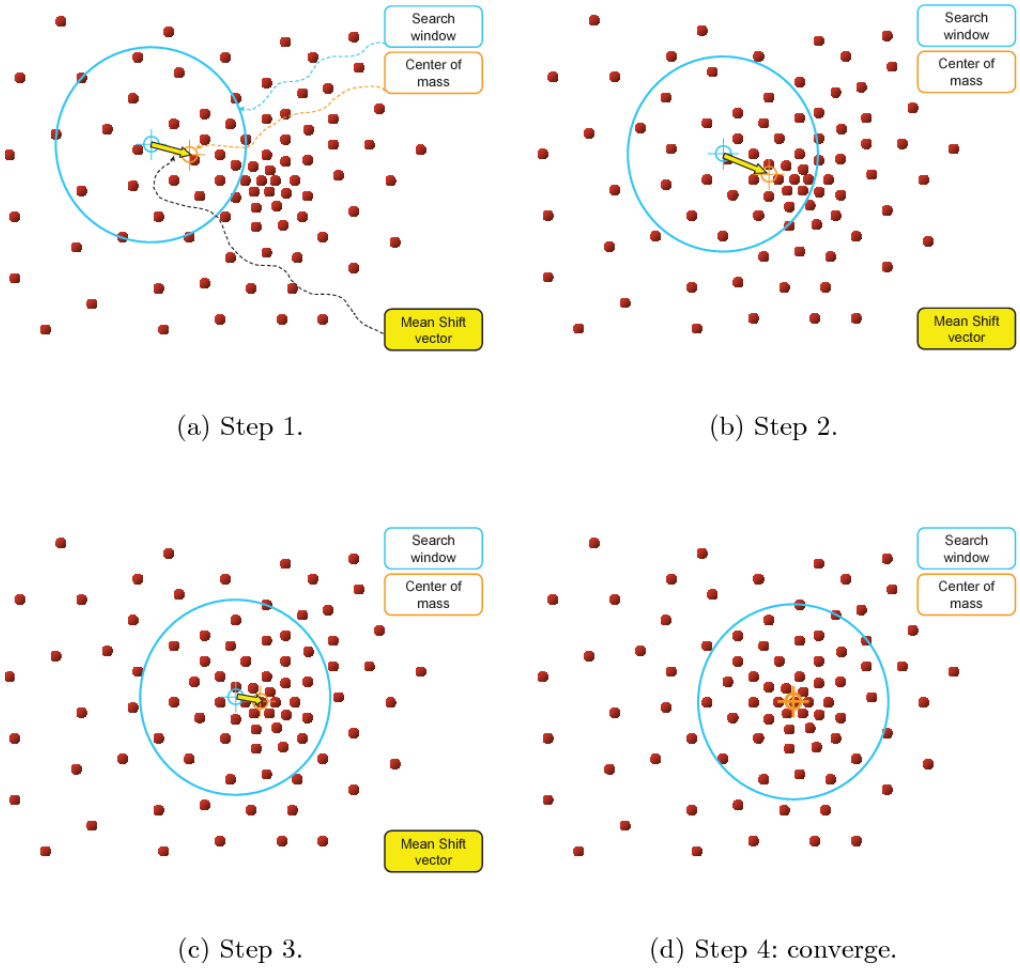
Са друге стране *K-Means* има неколико недостатака. Прво се мора дефинисати колико кластера постоји. Ово није увек тривијално и у идеалном случају би било пожељно да то одради сам алгоритам. *K-Means* такође почиње са случајним избором центара кластера и зато може да произведе различите резултате у различитим извођењима алгоритма. Стога резултати могу бити поновљиви и недостаје доследност. Остали алгоритми кластеризације су конзистентнији.

K-Medians је још један алгоритам кластеризације који се ослања на *K-Means*, само уместо прерачунавања централних тачака група користећи средњу вредност, користи средишњи вектор групе. Ова метода је мање осетљива на присуство *outlier*-а, али је много спорији за веће скупове података јер је потребно сортирање на свакој итерацији приликом израчунавања *Median* вектора.

2.7.2 Mean-shift

Mean-shift кластеризација је алгоритам заснован на принципу покретног прозора (енг. *Sliding-window*), који покушава да пронађе области на којима се налази густ распоред тачака. То је алгоритам базиран на центроидима, што значи да је циљ лоцирати средишње тачке сваке групе/кластера. Своди се на ажурирање кандидата за централну тачку да буде средња вредност у оквиру. Прозори који представљају кандидате се филтрирају у фази накнадне обраде, како би се уклонили дупликати, формирајући тако коначни скуп средишњих тачака и њихових одговарајућих група.

1. *Mean-shift* је посматран на дводимензионалном скупу тачака (Слика 2.12). Алгоритам започиње са кружним покретним прозором, који представља језгро (енг. *Kernel*), са центром у слободно изабраној тачки и одређеним радијусом. *Mean-shift* је алгоритам који укључује померање језгра итеративно према региону са најгушћим распоредом тачака, све док конвергира.
2. Са сваком итерацијом, клизни прозор се помера ка региону са већом густином, на основу померања његове средишње тачке која представља центар масе свих тачака унутар покретног прозора.
3. Померање покретног прозора се наставља до момента где центар масе свих тачака унутар прозора нема тенденцију да се помера.
4. Када се више покретних прозора преклапа, сачуван је прозор који садржи највише тачака. Подаци се затим групишу према прозору којем припадају.



(a) Step 1. (b) Step 2.

(c) Step 3. (d) Step 4: converge.

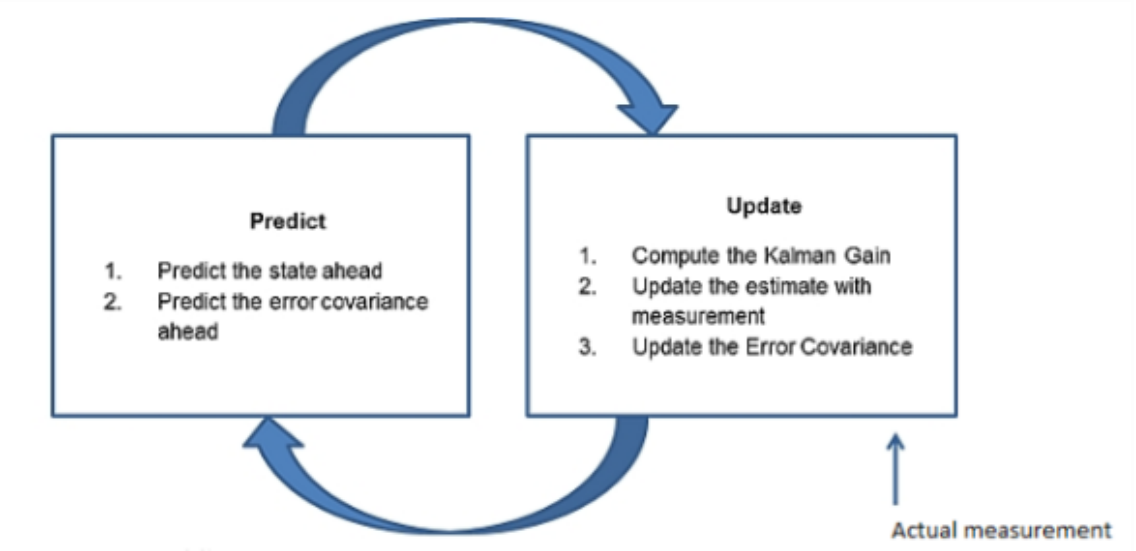
Слика 2.12 *Mean-shift* алгоритам кластеризације

У поређењу са *K-means* алгоритмом, нема потребе да се унапред дефинише број кластера, јер *Mean-shift* то аутоматски открива. Ово је значајна предност овог алгоритма. Чињеница да центри кластера конвергирају према тачкама максималне густине такође је прилично пожељна, јер је прилично интуитивно разумети. Недостатак представља чињеница да одабир величине прозора није тривијалан задатак.

2.8 Праћење објеката (Калман филтер)

Помоћу Калман филтера [12] се врши претпоставка, шта је то што ће посматрани систем урадити или у каквом стању бити у наредном моменту. Он се може употребити где год постоје неке несигурне информације о неком динамичком систему. У случају аутономног возила Калман филтер се може користити за предвиђање наредног низа акција које ће аутомобил испред нас предузети, на основу података које его возило прими преко сензора. То је итеративни поступак који користи два корака, предвиђање (енг. *Predict*) и ажурирање (енг. *Update*). Он је погодан за ову врсту операција на основу следећих особина.

- Предвиђа следеће стање само на основу претходног стања, без потребе за познавањем комплетне историје података.
- Рачунарски веома брзо решење, што га чини погодним за проблеме у реалном времену.
- Чак и у ситуацијама где постоји пуно шума у самом окружењу, решење ће дати добар резултат захваљујући Гаусовој расподели.



Слика 2.13 Поступак извршавања Калман филтера

2.8.1 Предвиђање (енг. *Predict*)

У овом кораку Калман филтер предвиђа нове вредности, а затим предвиђа грешку претпоставке у складу са различитим грешкама током процеса у систему. Код аутономних возила, грешка у процесу се може разумети једноставним примером аутомобила који се креће испред возила. Модел ће претпоставити да се аутомобил креће константном брзином због нултог убрзања, али у стварности ће имати убрзање, тј. брзина ће с времена на време флуктуирати. Промена брзине тог аутомобила ће представљати грешку, а то се систему представља као процесна грешка.

2.8.2 Ажурирање (енг. *Update*)

У овом кораку се узима стварна измерена вредност са уређаја. У случају аутономног возила тај уређај може бити камера, радар, *LIDAR* … Затим се израчунава разлика између предвиђене вредности и измерене вредности, а затим се одлучује која вредност се задржава. Ова одлука се доноси на основу Калманове добити (енг. *Kalman Gain*). Након одлуке добијене на основу Калманове добити, поново се израчунава нова вредност и нова грешка. Овај излаз из корака ажурирања се поново враћа у корак предвиђања и процес се наставља,

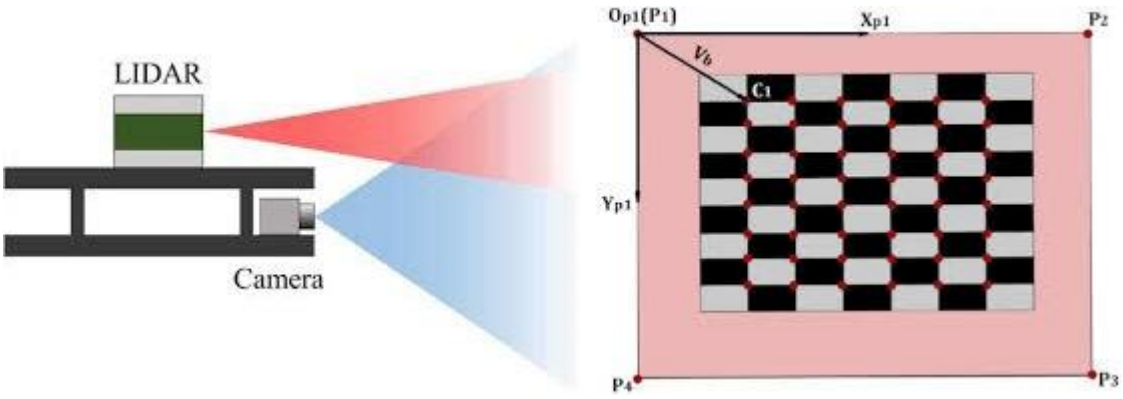
све док разлика између предвиђене вредности и измерене вредности тежи нули. Ова израчуната вредност представља претпоставку добијену од стране Калман филтера.

Калманова добит утврђује да ли је предвиђена или измерена вредност близу стварне вредности. Његова вредност се креће од 0 до 1. Ако је његова вредност близу нуле, то значи да је предвиђена вредност близу стварне вредности. У случају да је вредност Калманове добити близу јединице, тада значи да је крајња измерена вредност близу стварне вредности. Његова вредност се креће од 0 до 1 јер користи фактор несигурности у предвиђеној и измереној вредности.

2.9 Фузија сензора (калибрација)

Тродимензионалне технике мапирања и локализације за примену у аутономним возилима се истражују дужи низ година. Најчешће коришћена техника у ову сврху се своди на фузију више сензора из тог разлога што сваки од појединачних сензора поседује неке предности и неке мане. Акценат у овом поглављу ће бити на фузији камере и *LIDAR*-а [13], који су коришћени током израде решења. Информација коју добијамо са камере се одликује великом густином самих података (резолуција), као и информацијом о боји сваког појединачног узорка (пиксел). Са друге стране појединачна камера нема перцепцију дубине, те из тог разлога, чак и да објекат буде успешно детектован, не може се утврдити његова удаљеност од возила. У ту сврху користимо *LIDAR* чија је главна одлика перцепција дубине. Међутим *LIDAR* за разлику од камере има проблем са густином самих податак које генерише. Што су детекције удаљеније од возила број детектованих тачака је све мањи. Поред тога *LIDAR* не пружа информацију о боји. Успешном калибрацином ова два сензора, могуће је искористити предности сваког од ова два сензора и добити крајњу информацију од значаја, а то је, поред саме детекције и класификације неког објекта, одређивање и његове удаљености

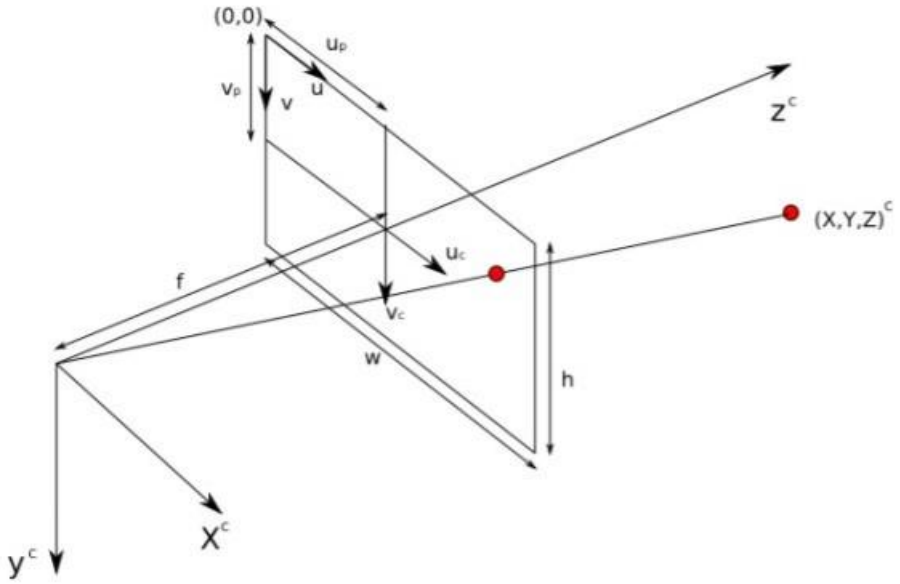
Постоји више начина како извршити фузију *LIDAR*-а и камере, како би се добио *3D* модел који има и компоненту боје. Однос података са *LIDAR*-а и са камере се користи за креирање трансформационе матрице. Како би се одредила трансформациона матрица преко које је могуће *3D* податке са *LIDAR*-а превести у *2D* простор, неопходно је исти објекат посматрати са оба типа сензора. То значи да оба сензора у истом тренутку треба да детектују исти предмет. Најчешће коришћен предмет у ову сврху јесте црно-бела шаховница (Слика 2.14).



Слика 2.14 Окружење за калибрацију

LIDAR добија *3D* податке о шаховници са информацијом о дистанци и интензитету. Са друге стране камера добавља информацију о боји и текстури шаховнице. Није могуће директно одредити којој *3D* тачки припада који *2D* пиксел. Након конверзије *LIDAR* података неопходно је наћи однос тачка на тачку (енг. *Point-to-point*), између *2D* података са *LIDAR*-а и података са камере. Однос се проналази екстраховањем тачака од значаја (ивице и углови) са шаховнице. Тачке од значаја је лако детектовати на оквиру (енг *Frame*) са камере, док се тачке од значаја на подацима са *LIDAR*-а добављају ручно, услед слабијег интензитета информације са *LIDAR*-а. У оквиру *ROS* окружења, коришћеном током израде овог рада, постоји пакет који омогућава мануелно означавање тачака од значаја. Процес је неопходно поновити при посматрању шаховнице из више различитих углова.

LIDAR 3D подаци су представљени у координатном систему света, из тог разлога је неопходно наћи везу између координатног система света и координатног система камере. За координатни систем који пројектује камера (Слика 2.15), раван слике је позиционирана дуж оптичке осе са одређеном жижном даљином. Принцип пројекције камером служи за конвертовање *3D LIDAR* тачака у *2D LIDAR* слику. *XYZ* координате представљају координате света. *Uc* и *Vc* су координате тачака у координатном систему света, које су пројектоване у *2D* раван. А раван је за жижну даљину удаљена од оригиналне позиције. *Z* представља оптичку осу, док је *X* и *Y* раван паралелна са оптичком равни. Правилом о једнакости троуглова, може се одредити веза између координатног система слике и координатног система света и добити коначна пројекција перспективе.



Слика 2.15 Принцип пројекције камером

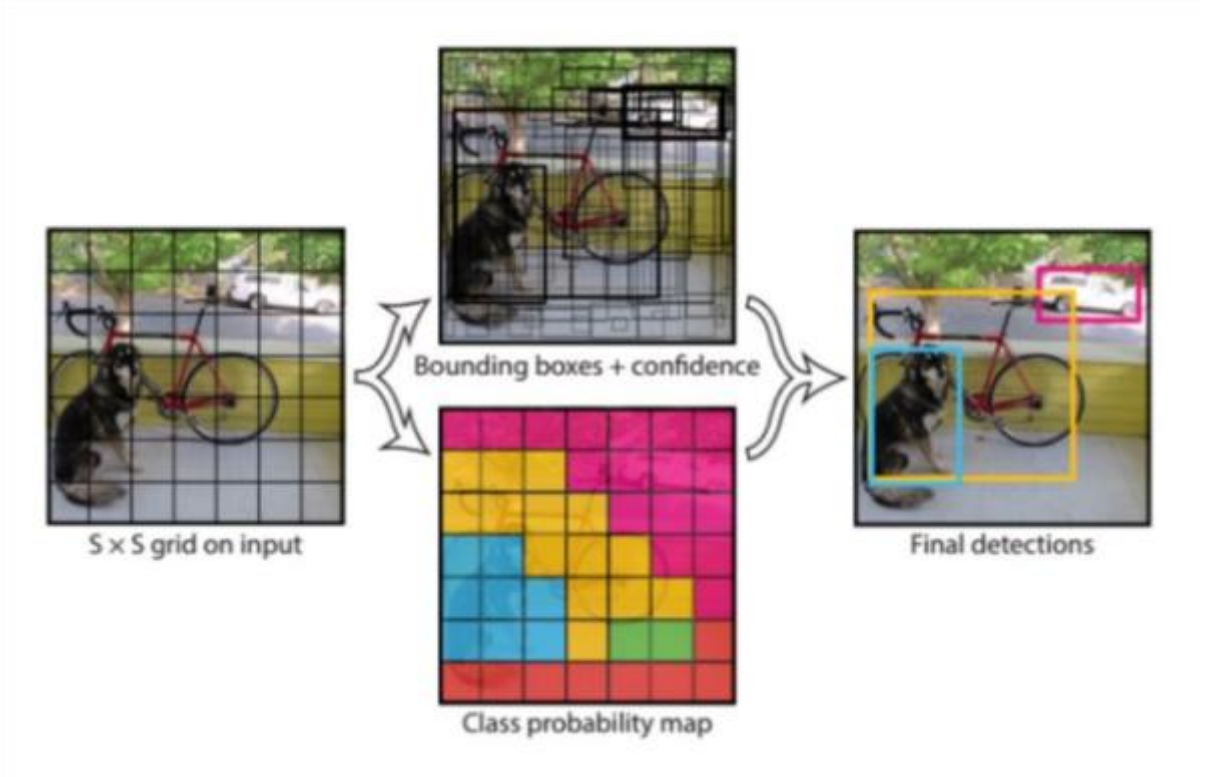
2.10 *2D* детекција објеката коришћењем *YOLO* неуронске мреже

Детекција објеката је један од класичних проблема у компјутерској визији, који има за циљ да одговори на питања шта представља детектовани објекат, као и то где се он налази на слици. То је управо оно шта *YOLO* ради, поред проблема класификације *YOLO* решава и проблем локализације. *YOLO* [14] је једна од најбржих неуронских мрежа за обраду слике и то је један од главних разлога за избор баш ове мреже у сврху извршавања у реалном времену. На неким *GPU*-овима може се постићи брзина обраде и до 45 *FPS*-а (*Frames Per Second*). Разлог зашто је ова мрежа тако брза лежи у чињеници да она обавља класификацију и локализацију једним проласком, док многе друге неуронске мреже користе два пролаза. Оно што је још једна значајна карактеристика *YOLO*-а и што му даје предност у односу на остале конволуционе неуронске мреже је то што се може балансирати између брзине и прецизности, једноставном променом величине модела. Код *YOLO*-а једна конволуциона неуронска мрежа предлаже више граничних оквира (енг. *Bounding boxes*) и вероватноће припадности одређеној класи за те предложене оквире

2.10.1 Принцип функционисања

YOLO дели улазну слику на мрежу величине 13x13 ћелија. Свака од тих ћелија је одговорна за предлагање 5 граничних оквира. Гранични оквир представља правоугаоник који уоквирује предложени објекат. *YOLO* такође на свом излазу даје информацију о томе колико је заиста сигуран да креирани гранични оквир заиста уоквирује реалан објекат. Овај

резултат не говори о томе којој врсти/класи припада детектовани објекат, већ о томе да ли је креирани гранични оквир заиста прецизно позициониран и да је одговарајуће величине. Предложени гранични оквири изгледају као на следећој слици (Слика 2.16), где су гранични оквири са већом вероватноћом сигурности исцртани дебљом линијом.

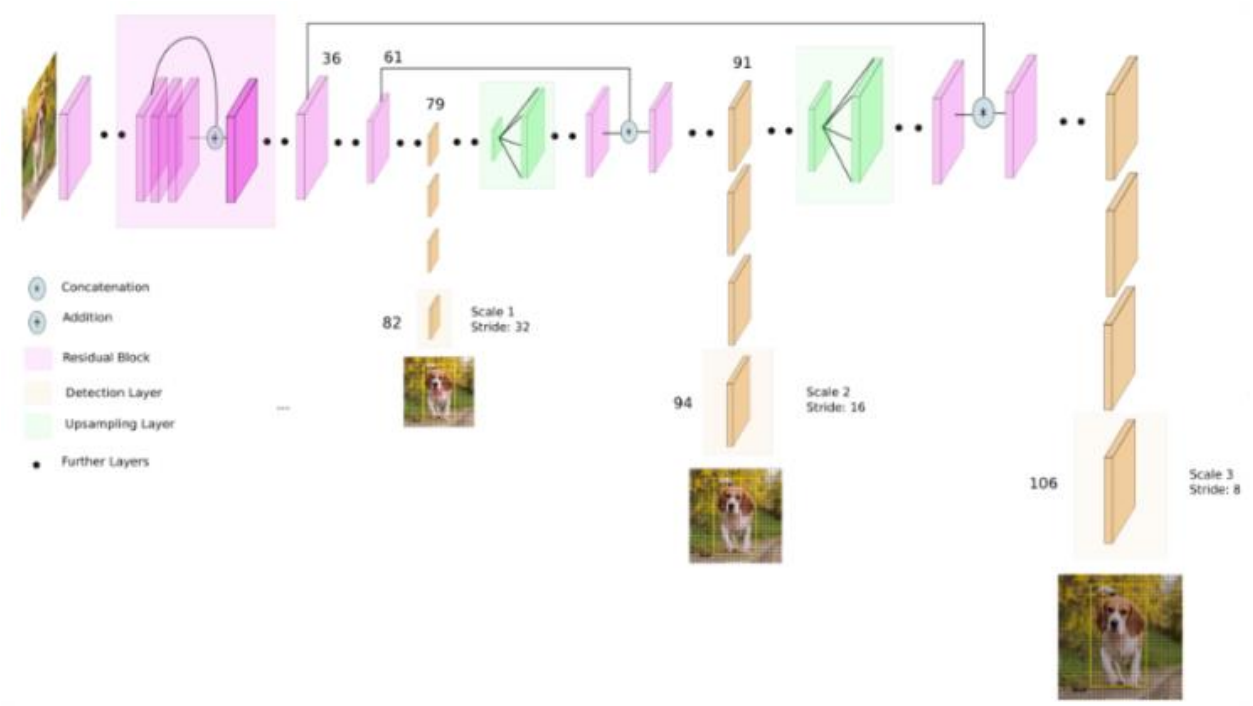


Слика 2.16 *YOLOv2* детекција објекта на слици

За сваки од граничних оквира, ћелија предлаже класу. Ово функционише као класичан процес класификације, он даје дистрибуцију вероватноће за све могуће класе. Резултат који представља сигурност да је гранични оквир успешно позициониран, као и информација о претпоставци припадности одређеној класи, заједно се комбинују у коначни резултат који говори о вероватноћи да се у детектованом граничном оквиру налази објекат одређене класе.

С обзиром да постоји 13x13 = 169 ћелија и свака ћелија предлаже 5 граничних оквира (*YOLOv2*), долазимо до бројке од 845 граничних оквира. Испоставља се да већина тих граничних оквира има веома низак резултат о уверености, зато се задржавају само оквири чији је крајњи резултат изнад дефинисане вредности. Та вредност се дефинише у зависности од тога колико желимо да детекције буду прецизне. Поступак је приказан на наредној слици (Слика 2.17).

2.10.2 Детекција на три нивоа

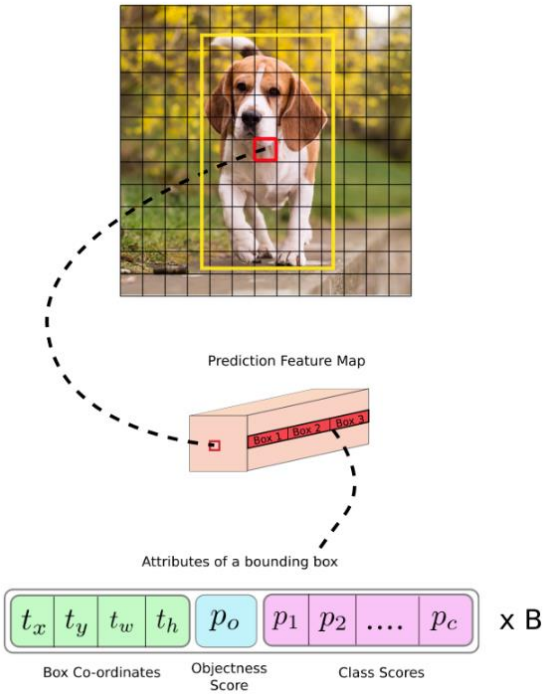


Слика 2.17 Архитектура *YOLOv3* неуронске мреже

Ова неуронска мрежа користи стандардне типове слојева: конволуциони слој са 3x3 језгром (енг. *Kernel*) и *max-pooling* са 2x2 језгром. У верзији 2 не постоје потпуно повезани слојеви (енг. *Fully connected layers*), док они постоје у верзији 3.

Нова верзија [15] је надограђена резидуалним блоковима и слојем за уметање (енг. Up sampling). Међутим, најизраженија карактеристика верзије 3 је вршење детекције на три различита нивоа, који су представљени умањивањем димензија улазне слике за 32, 16 и 8 респективно.

Оно што још верзија три уводи јесте променљив број детекција за сваку од ћелија. Док је код *YOLOv2* овај број био фиксиран на 5, код *YOLOv3* то није случај. Тако да сваки предложак од стране једне ћелије изгледа следеће.



Слика 2.18 Предложак за појединачну ћелију *YOLOv3*

tx, ty, tw, th – позиција и величина предложеног граничног оквира

Po – Вероватноћа да се у граничном оквиру заиста налази објекат

P1, …, Pc – Вероватноће припадности одређеној класи

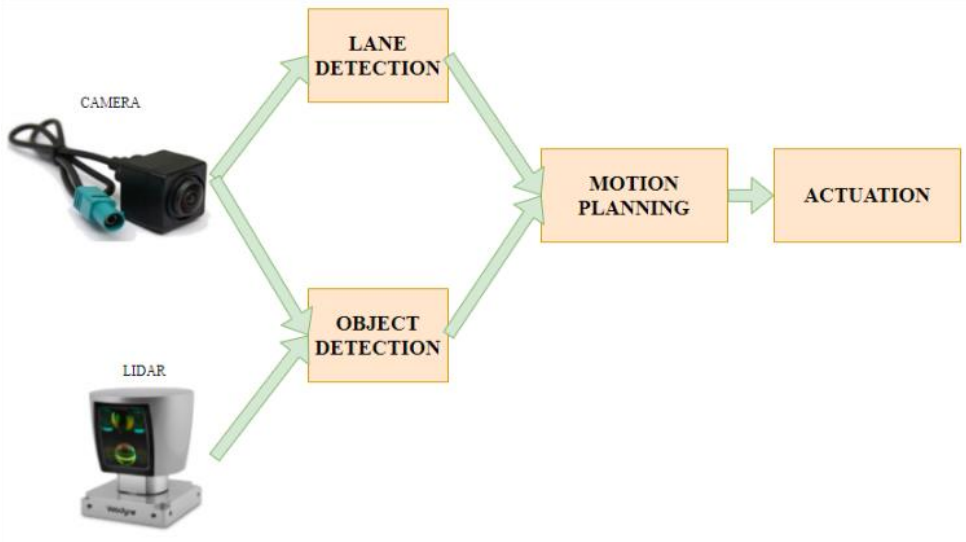
2.10.3 Ограничења *YOLO* мреже

YOLO намеће строга просторна ограничења током предвиђања граничних оквира, јер свака ћелија решетке предвиђа само два поља и може имати само једну класу. Ово просторно ограничење намеће проблем детектовања више малих објеката који се налазе у непосредној близини (нпр. Јато птица). Наш модел такође користи релативно грубе функције за предвиђање граничних поља, јер наша архитектура има више слојева за смањивање узорака са улазне слике. Током тренинга се перформансе самих детекција побољшавају на основу података који се добијају од стране функција губитака (енг. *Loss functions*), међутим функција губитка третира подједнако грешке, како у малим тако и у великим граничним оквирима. Мала грешка код великог оквира је углавном безазлена, али мала грешка код малог граничног оквира има много већи утицај на *IOU* (*Intersection Over Union*). Главни извор грешке је погрешна локализација.

3. Решење

У оквиру овог поглавља биће представљено једно решење за детекцију објекта који се налази непосредно испред возила, као и процена његове удаљености, у сврху имплементације система за спречавање чеоног судара. Решење ће бити претходно стављено у шири контекст, где је за циљ овај пројекат имао реализацију система који имплементира ниво 2 аутономне вожње. Поред поменутог *CAS* система, на којем је овде акценат, имплементиран је и систем за одржавање возила унутар траке (енг. *Lane Keeping System*). Начин на који су концепти, приказани у поглавља „Теоријске основе“, спрегнути у циљу имплементације алгоритма за процену удаљености до препреке, биће детаљно објашњено овде.

3.1 Преглед комплетног система

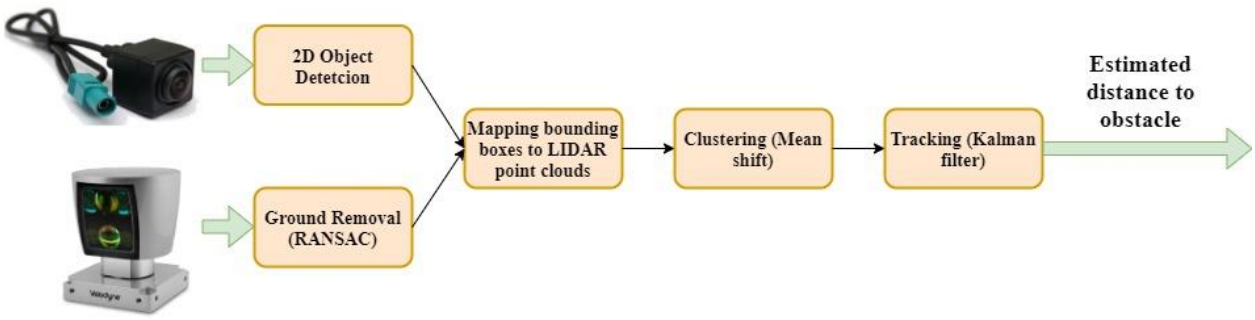


Слика 3.1 Блок шема комплетног система

Комплетан систем (Слика 3.1) који је развијан у оквиру овог пројекта имплементира две главне функционалности. Прва функционалност представља систем за одржавање возила унутар возне траке (енг. *Lane Keeping System*). Другу функционалност представља систем за избегавање колизије са препреком које се налази непосредно испред возила *CAS*. Као улаз подсистем за одржавање возила унутар траке користи се слика која стиже са камере, док *CAS* подсистем користи податке како са камере тако и са *LIDAR*-а. Модул за детекцију траке којом се возило креће (енг. *Lane Detection*) и модул за детекцију објекта које се налази непосредно испред возила (енг. *Object Detection*) шаљу потребне информације модулу за планирање кретања (енг. *Motion Planning*), који на основу прослеђених информација дефинише параметре актуације, које задаје модулу за актуацију (енг. *Actuation*). На основу података које модул за планирање прими од модула за детекцију траке којом се возило креће, одређују се параметри који дефинишу латералну кретњу возила. Са друге стране податак о удаљености од евентуалне препреке, који долази од стране модула за детекцију објеката дефинише параметре везане за лонгитудиналну кретњу возила. На основу тих параметара, модул за актуацију реагује на критичне догађаје, као што су евентуално напуштање возне траке или критична раздаљина до возила испред, и шаље команде директним хардверским склоповима, као што су волан, кочница или папучица гаса.

3.2 Концептуални опис решења

Процес планирања акције и актуације неће бити објашњени у овом поглављу, нити је акценат приликом имплементације био на томе. Оно што је овде од интереса, јесте начин детекције објекта од интереса, као и процена његове удаљености. Начин на који је то изведено илуструје следећа слика (Слика 3.2).



Слика 3.2 Блок шема подсистема за детекцију и процену удаљености објекта

У ову сврху су коришћени подаци са два различита типа сензора. *2D* подаци добијени са монокуларне камере и *3D* подаци добијени са *LIDAR*-а. Идеја јесте да се *RGB* подаци пристигли са камере користе у сврху детекције и класификације објеката, из тог разлога што већ постоје бројни алгоритми машинског учења, који се баве *2D* детекцијом објеката.

Алгоритам који је коришћен у оквиру овог рада у ту сврху јесте *YOLOv3*. Са друге стране услед недостатка перцепције дубине коришћен је *LIDAR* са својим *3D* скупом података, који служи да да информацију о удаљености детектованог објекта. Прва ствар коју је неопходно одрадити приликом коришћења ова два сензора, јесте њихова калибрација. Идеја калибрације у овом случају јесте да се скуп података са *LIDAR* сензора доведу у перспективу из које камера посматра околину. Да се *3D* подаци са *LIDAR*-а мапирају на *2D* простор камере. Поента свега овога јесте да се одреди, који објекат детектован на оквиру (енг. *Frame*) добијена са камере, одговара којем кластеру *3D* тачака са *LIDAR*-а.

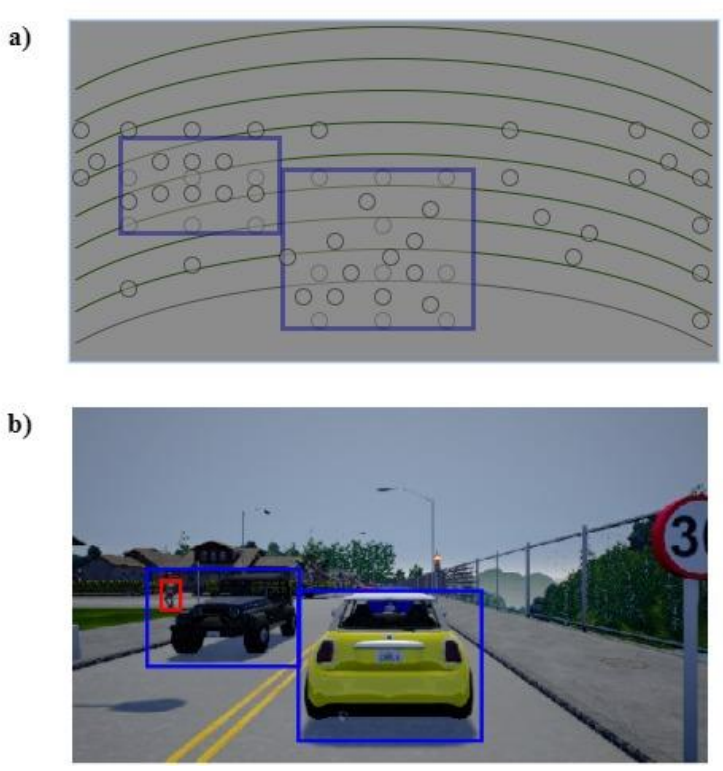
У *ROS* окружењу коришћеном током израде овог рада постоји уграђени пакет за калибрацију сензора [16]. *LIDAR*-камера калибрација се своди на кликтање на кореспондентне тачке у оквиру камера оквира и *LIDAR* облака података. Поред описане генералне процедуре калибрација, у оквиру ROS окружења постоји имплементиран пакет, који поједностављује сам процес калибрације.

Пре него што се обави мапирање *3D* облака података у *2D* простор камере, потребно је уклонити што више редудантних података. У ову групу података спадају све тачке за које се у старту зна да неће припадати објекту од значаја. У идеалном случају једине релевантне тачке би биле оне које се налазе у возној траци којом се креће возило, са изузетком оних тачака које припадају путу. Имплементирано решење нема модул за детектовање возне траке у смислу *3D* података, пред-обрада је одрађена са нешто мањом прецизношћу. Оно што се у овом решењу одстрањује на самом почетку, јесу подаци који се налазе довољно далеко од возила, непосредно око самог сензора (тачке које припадају нашем возилу), тачке које се налазе изнад одређене висине и као најбитније тачке које припадају површини пута. Филтриране тачке се налазе илустроване на наредној слици (Слика 3.3).

Слика 3.3 Пред-процесирање облака података са *LIDAR*-а

Радијус са којим се жели извршити филтрирање може да се дефинише по потреби. Главни део филтрирања јесте филтрирање површине пута. У ову сврху је одлучено коришћење *RANSAC* алгоритма, из тог разлога што је релативно једноставан за имплементацију, а при томе је и брз. Постоје и мане по питању недетерминизма овог алгоритма и проблема приликом детекције пута у случају да пут није на свим местима исто нивелисан, међутим за неке уобичајене случајеве ово решење ради на задовољавајућем нивоу.

Након што су исфилтрирани подаци од значаја, *3D* подаци се на основу параметара добијених процесом калибрације преводе у *2D* простор који одговара видном пољу камере. У овом моменту постоје детектовани гранични оквири на оквиру добијеног са камере и *2D* пројекцију облака података са *LIDAR*-а. Након тога је могуће, на основу обичног позиционог мапирња, установити који облаци података припадају којем граничном оквиру детектованом са камере (Слика 3.4).



Слика 3.4 Мапирање *2D* граничних оквира на *3D* облаке података

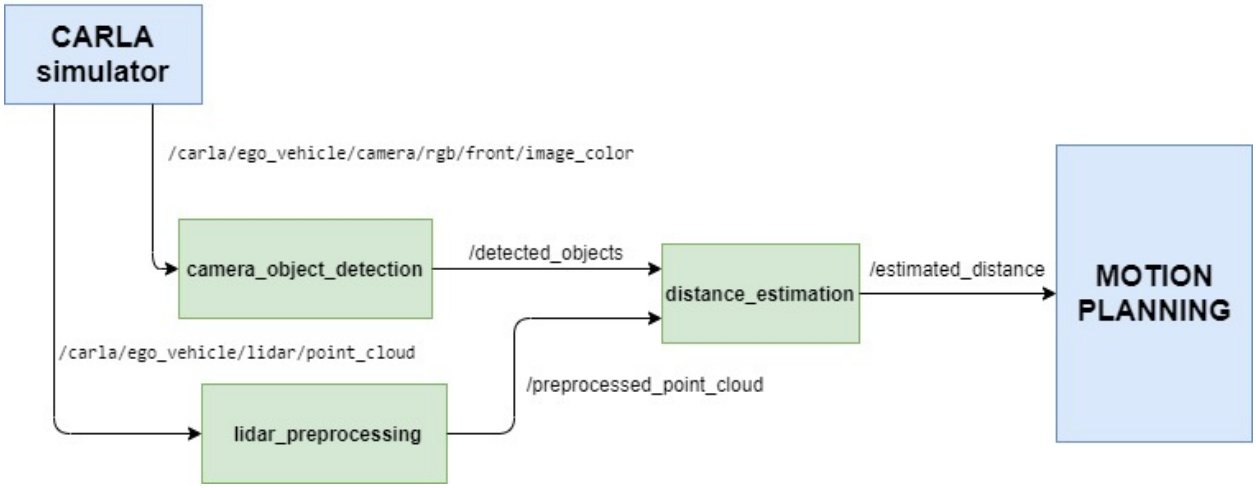
Следећи корак јесте поделити све облаке података који припадају различитим граничним оквирима у кластере. За сваки од граничних оквира може постојати највише три различита кластера, који се деле по просторној дистрибуцији. Кластер се прави на основу вредности координате која представља дубину. Та три кластера представљају позадину, први план и сам објекат. Међутим често се догоди да не постоји предњи план, и зато се не

може знати унапред колики број кластера треба одредити. Из тог разлога је дата предност *Mean-shift* алгоритму за кластеризацију у односу на *K-means*. Овај поступак се понавља за сваки од детектованих граничних оквира. Оно на шта треба обратити пажњу јесте да се неки од ових кластера могу понављати међу различитим граничним оквирима. Кластер који представља сам објекат унутар једног граничног оквира може представљати предњи план унутар другог граничног оквира. То се решава на основу координата централних тачака детектованих кластера, где се дупликати одстрањују.

Након овог корака и после кластеризације који је објашњен у претходном поглављу, одређени су кластери који представљају објекте од значаја као и њихове централне тачке. Једини преостали корак јесте откривање баш оног објекта који је од значаја (возило које се креће непосредно испред его возила, у истој возној траци). Због једноставности, за тражени објекат бирамо онај чија се централна тачка по координатном систему *LIDAR*-а налази непосредно испред его возила.

Након овога је добијена информација о удаљености до препреке непосредно испред его возила. Ову информацију модул за детекцију објекта даље шаље модулу за планирање кретања, како би дефинисао потребне акције уколико постоји потреба за тим (евентуална колизија). Као последња ставка, која служи да се предвиди позиција препреке у наредном тренутку, јесте Калман филтер. Он се користи за предвиђање будућег стања система који се надгледа (возило непосредно које се креће непосредно испред), а погодан је за ситуације у којима постоји несигурна информација о неком динамичком систему.

3.3 Програмско решење



Слика 3.5 Архитектура програмског решења

Као што је речено, током израде коришћено је *ROS* окружење. У складу са тим, функционални блокови су имплементирани у форми чворова (енг. *Node*). Комплетна

имплементација детекције објекта и процена његове удаљености је издељена у три чвора, као што је приказано на слици (Слика 3.5).

CARLA симулатор на основу *ROS* моста (енг. *ROS Bridge*) своје интерне сигнале које добија са различитих сензора емитује у форми *ROS* порука, које се објављују на одговарајућој магистрали/теми (енг. *ROS Topics*). Сензори који су додати на његово возило у ову сврху јесу једна предња монокуларна камера и *LIDAR*. Сваки од ова два сензора емитује податке на своју тему, са одговарајућом структуром, који дају информације о окружењу. Модул за детекцију објеката са камере (енг. *CAMERA object detection*) је претплаћен на тему под називом „*/image_color*“, која у оквиру себе садржи поруку са следећим форматом (Слика 3.6).

```
std_msgs/Header header
uint32 height
uint32 width
string encoding
uint8 is_bigendian
uint32 step
uint8[] data
```

Слика 3.6 Формат поруке добијен са камера сензора

Овакав формат поруке се са фреквенцијом од 30 *fps*-а доводи на улаз за детекцију објекта. У сврху имплементације у складу са *ROS* окружењем коришћен је *Python*-ов *ROSPY* модул. У склопу овог модула постоји функција „*Subscriber*“, која служи да се чвор претплати на одређену тему.

```
from sensor_msgs.msg import Image

def image_cb(msg):
    img = bridge.imgmsg_to_cv2(msg, "bgr8")
    img_box = yolo(img)
    img_box_msg = bridge.cv2_to_imgmsg(img_box, encoding="passthrough")
    pub =rospy.Publisher('detected_objects', Image)
    pub.publish(img_box_msg)


def camera_object_detection():
    rospy.init_node('camera_object_detection', anonymous = True)
    camera_sub = rospy.Subscriber('carla/ego_vehicle/camera/rgb/front/image_color', Image, image_cb)

    rospy.spin()
```

Слика 3.7 Имплементација *ROS* чвора који обавља *2D* детекцију објеката на улазној слици

За сваки од пристиглих оквира се окида функција „*image_cb*“ у оквиру које се позива *YOLO* алгоритам за *2D* детекцију објеката (Слика 3.7). Након што се добију детекције објеката у виду граничних оквира (енг. *Bounding boxes*), оваква слика се прослеђује на улаз

чвора за процену удаљености (енг. *Distance estimation*). Формат излазне поруке, која се прослеђује у оквиру теме под називом „*detected_objects*“ остаје исти као и улазна порука.

Са друге стране модул за предпроцесирање података са *LIDAR*-а прима податке у оквиру теме „*/point_cloud*“ која садржи поруку следећег формата (Слика 3.7).

```
std_msgs/Header header
uint32 height
uint32 width
sensor_msgs/PointField[] fields
bool is_bigendian
uint32 point_step
uint32 row_step
uint8[] data
bool is_dense
```

Слика 3.8 Формат поруке добијен са *LIDAR* сензора

Једна од битнијих ствари коју је неопходно одрадити приликом прикупљања података са *LIDAR* сензора, јесте одстрањивање редудантних података. Оно што је било представљено у самом опису решења јесте одстрањивање податак, за које је у старту познато да не припадају траженом објекту и на тај начин се смањује скуп податка за главну фазу обраде у оквиру које је неопходно пронаћи кластер тачака, који припада траженом објекту и одредити раздаљину до њега.

```
from sensor_msgs.msg import PointCloud2

def lidar_cb(msg):
    preprocessed_msg = filter_by_radius(msg)
    preprocessed_msg = filter_by_height(preprocessed_msg)
    preprocessed_msg = filter_ground(preprocessed_msg)
    pub = rospy.Publisher('preprocessed_point_cloud', PointCloud2)
    pub.publish(preprocessed_msg)

def lidar_preprocessing():
    rospy.init_node('lidar_preprocessing', anonymous = True)
    camera_sub = rospy.Subscriber('carla/ego_vehicle/lidar/point_cloud', PointCloud2, lidar_cb)

    rospy.spin()
```

Слика 3.9 Имплементација *ROS* чвора за пред-обраду улазних података са *LIDAR* сензора

Оно што се може закључити из приложеног кода (Слика 3.9), као што је и објашњено у самом опису решења, филтрирање података са *LIDAR*-а се обавља у три фазе. Подаци се филтрирају прво по унапред дефинисаном радијусу, где се за релевантне тачке узимају само подаци који су у кругу од 30 метара од возила. Потом се преостали подаци филтрирају по висини, чиме се одстрањују све тачке које припадају објектима који се налазе изнад пута и не представљају препреку самом возилу (нпр. крошња дрвета, надвожњак …). Као

незаобилазни вид филтрирања, када је реч о подацима пристиглим са *LIDAR*-а, позива се функција за уклањање података који припадају површини пута. Као што је већ поменуто, у ову сврху је коришћен *RANSAC* алгоритам. Ове три функције за уклањање редудантних података су имплементиране у оквиру засебног *Python* модула. Након што су одстрањени сви редуданти подаци пристигли са *LIDAR* сензора, за које постоји сигурност да не припадају траженом објекту, овај чвор објављује тему под називом „*preprocessed_point_cloud*“, која садржи исти формат поруке као и улазна тема.

Као последњи ниво обраде, у оквиру којег је имплементирана преостала логика за детекцију траженог објекта, као и процену његове удаљености, имплементиран је чвор под називом „*distance_estimation*“ (Слика 3.10).

```
from sensor_msgs.msg import Image
from sensor_msgs.msg import PointCloud2

img_ready = False
lidar_point_cloud_ready = False
img = None
lidar_point_cloud = None


def distance_estimation():
    while not rospy.is_shutdown():
        if img_ready and lidar_point_cloud_ready:
            # this call returns 2d representation of lidar point cloud based on sensor calibration
            2d_lidar_point_cloud = projection_of_lidar_point_cloud_to_2d(img, lidar_point_cloud)
            # this call returns lidar point cloud that belong detected bounding boxes
            detected_point_cloud = mapping_bounding_boxes_to_lidar_point_cloud(img, 2d_lidar_point_cloud)
            # this call returns cluster that belongs to target object
            target_object_point_cloud = mean_shift_clustering(detected_point_cloud)
            # this call returns estimated distance of target object
            estimated_distance = kalman_filter(target_object_point_cloud)

            pub = rospy.Publisher('/estimated_distance', EstimatedDistance)
            pub.publish(estimated_distance)

            img_ready = False
            lidar_point_cloud_ready = False


def image_cb(msg):
    global img = bridge.imgmsg_to_cv2(msg, "bgr8")
    img_ready = True

def lidar_cb(msg):
    global lidar_point_cloud = msg.data
    lidar_point_cloud_ready = True

def main():
    thread = threading.Thread(target=distance_estimation)
    thread.start()

    rospy.init_node('distance_estimation', anonymous = True)
    camera_sub = rospy.Subscriber('/detected_objects', Image, image_cb)
    lidar_sub = rospy.Subscriber('/preprocessed_point_cloud', PointCloud2, lidar_cb)

    rospy.spin()
```

Слика 3.10 Имплементација *ROS* чвора за процену раздаљине до препреке

Овај модул као улазне параметре прима слику са детектованим објектима у оквиру теме под називом „*/detectected_object*“, међу којима је неопходно пронаћи онај који је од значаја (возило које се креће непосредно испред его возила) и исфилтриране податке са *LIDAR* сензора у форми облака података (енг. *Point cloud*) који се шаље у оквиру теме под називом „*preprocessed_point_cloud* “. Основни костур кода, који имплементира овај модул, приказан је у наставку.

Временска синхронизација између података који пристижу са различитих сензора, није подржана у оквиру овог решења, али њен утицај је разматран у поглављу „Поступак испитивања и резултати“. Овде је овај поступак поједностављен и заснива се на томе да функција за процену удаљености траженог објекта чека да чвор прими обе поруке и тек се онда креће са поступком обраде. Обрада се заснива на секвенцијалном позивању четири функције за процесирање пристиглих података.

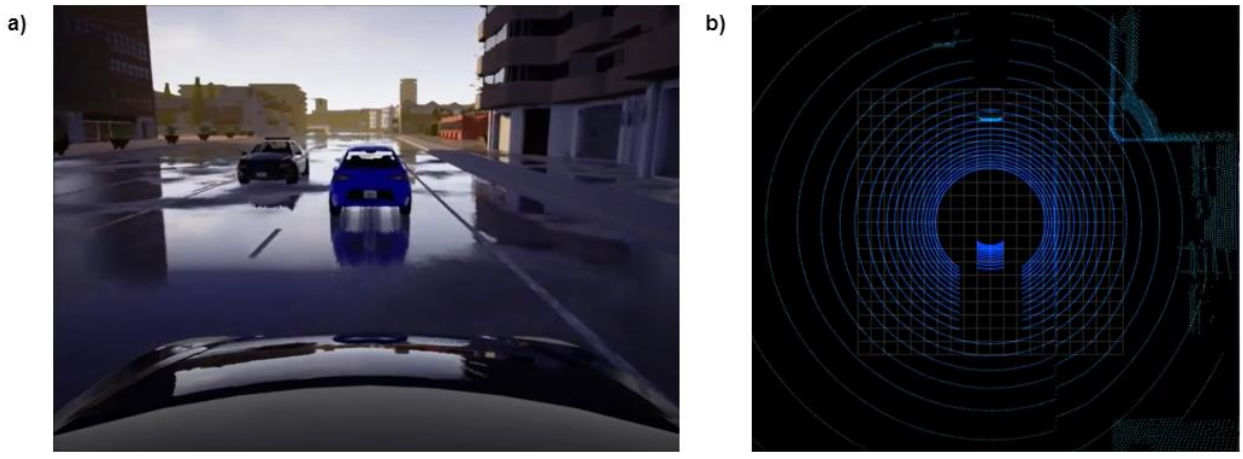
1. ***projection_of_lidar_point_cloud_to_2d*** – Ова функција као параметре прима пристигли снимак са камере са детектованим објектима и пред-обраде податке са *LIDAR* сензора. На основу параметара калибрације у оквиру ове функције се *3D* подаци са *LIDAR* сензора транслирају у *2D* простор и таква информација се враћа као повратна вредност.
2. ***mapping_bounding_boxes_to_lidar_point_cloud*** – Ова функција као параметре прима слику са детектованим објектима и *2D* репрезентацију података са *LIDAR* сензора. Обичним мапирањем кореспондентних тачка, добија се облак података који припада детектованим граничним оквирима. Овај облак података се доводи на улаз алгоритма за кластеризацију.
3. ***mean_shift_clustering*** – Овај алгоритам као параметар прима облак података, који припада свим детектованим објектима. Оно што овај алгоритам одради јесте проналажење кластера тачака који припадају конкретним објектима, а не евентуално некој позадини или првом плану. На основу просторне расподеле овај алгоритам одређује баш онај кластер који припада траженом објекту (непосредна препрека унутар возне траке којом се креће его возило). Информација о кластеру који припада траженом објекту се прослеђује последњој функцији у низу, која користи Калман алгоритам, како би пратила кретање и дистанцу до препреке.
4. ***kalman_filter*** – Након што је детектован кластер, који припада траженом објекту, који представља референтни објекат за спречавање евентуалне колизије, Калман алгоритмом се прати његово кретање и информација о његовој дистанци се објављује на тему под називом „*/distance_estimation*“, у

оквиру које се прослеђује порука која садржи информацију о дистанци до детектоване препреке.

Након овога чворови, који су имплементирани у оквиру модула за планирање, могу да се претплате на ову тему и врше задавање команди актуаторима, како би се спречила евентуална колизија са објектом који се налази непосредно испред возила.

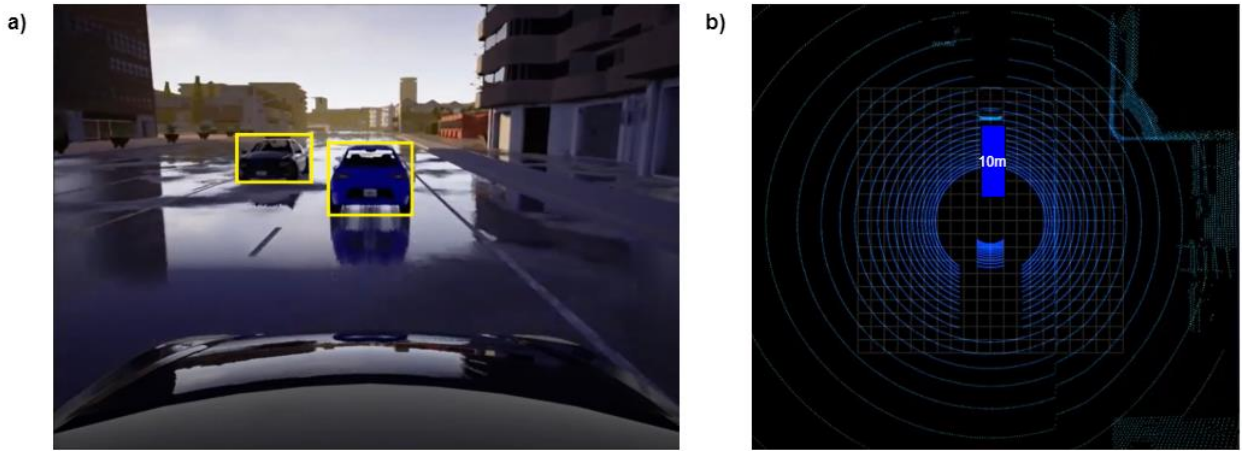
4. Поступак испитивања и резултати

Поступак испитивања је спроведен у симулираном окружењу (*CARLA* симулатор), услед мањег ризика и једноставности. Унутар симулатора постоје разне опције које омогућавају да се репродукују неке ситуације у саобраћају које су ретке у стварном животу или их је ризично изазвати у сврху тестирања. Поступак испитивања се базира на успешност детекције возила испред и избегавање колизије. Успешност детекције и процена удаљености је испитана у различитим временским условима, у разним временским тренуцима у току симулације. На слици испод (Слика 4.1) су приказани подаци које имплементирани алгоритам добија са два типа сензора (камера и *LIDAR*), унутар *CARLA* симулатора.



Слика 4.1 Улазни подаци у алгоритам

Визуелизацијом података, које добијемо на излазу система за детекцију и процену удаљености возила које се креће испред, у истој возној траци, изгледа као на следећој слици (Слика 4.2).



Слика 4.2 Визуелизација излазних података из алгоритма

Претходна слика илуструје резултате детекције возила које се креће непосредно испред. На основу фузије података са камере и *LIDAR*-а, алгоритам је успео да релативно успешно изврши процену удаљености детектованог возила. На основу података о позицији сваког учесника симулације, који се могу добавити из самог симулатора, вршено је поређење стварне и процењене дистанце до детектованог објекта. Наредна табела (Табела 4.1) приказује успешност процене алгоритма.

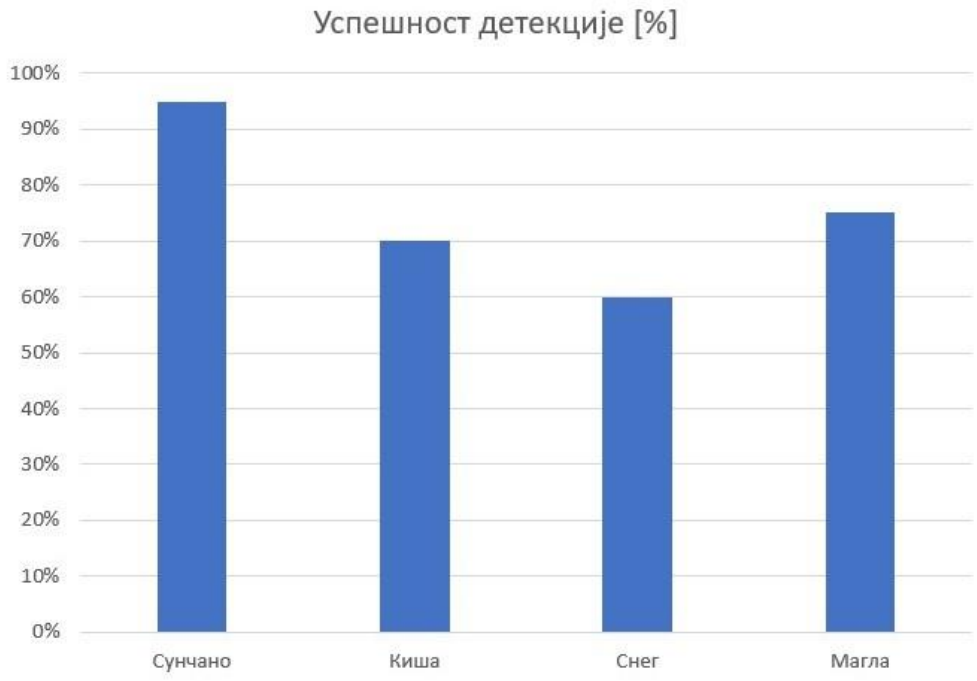
Редни број мерења	Процена алгоритма	Стварни резултати
1	40m	30m
2	28.6m	28.2m
3	25.3m	25m
4	23.9m	23.6m
5	21.5m	21.3m
6	30.5m	19.7m
7	18.7m	18.5m
8	18m	17.8m
9	17m	16.9m
10	16m	15.9m

Табела 4.1 Успешност процене раздаљине

Оно што има највећи фактор, на прецизност детекције јесте, одабир правог кластера облака података. Када се пронађе одговарајући кластер то је то, не би требало да постоји

нека значајна грешка у процени. На крају крајева, грешка која је прихваћена у оквиру овог решења, јесте услед одлуке да се за процену удаљености возила користи централна тачка кластер, а не најближу. Ситуација где се јавља највећи проблем јесте у случају да се погрешно процени кластер који се посматра. Ова ситуација се јавила пар пута, из тог разлога што алгоритам за тражени кластер узима онај највећи у оквиру детектованог граничног оквира. Проблем настаје када детектовани гранични оквир није довољно прецизан, па се некад деси да позадину узме као тражени објекат уместо самог возила (Табела 1. рб. 1). Такође проблем настаје у случају да је препрека удаљенија, где слаба густина облака података утиче на успешност детекције одговарајућег кластера (Табела1. рб. 6). Међутим овакви догађаји су појединачни и ретки, те из тог разлога у оквиру "*Motion planning*" модула се уз помоћ Калман филтера регулише овај проблем и у случају да резултат измерене дистанце јако одскаче од претходне вредности, нови резултат неће бити узет у разматрање при рачунању евентуалне актуације.

Током испитивања су коришћени разни временски услови, где се показао један недостатак оваквог вида фузије камере и *LIDAR*-а. Резултат процене дистанце до препреке је превише зависан од саме *2D* детекције са камере. Уколико *2D* детекција објекта није у стању да се успешно изврши услед неких ванских услова (нпр. Слаба видљивост), алгоритам неће бити у стању да детектује препреку. Обављено је двадесет тестова за сваки од четири различита временска услова, а успешност добијених детекција је приказана на дијаграму испод.



Слика 4.3 Успешност детекције

Чињеница јесте да је ово велики недостатак и да би неко проширење овог рада требало да уведе решавање овог проблема. Нека идеја јесте да се на неку временску периоду обрађују само подаци са *LIDAR*-а и да се независно од тога да ли је камера успела нешто да детектује, *LIDAR* провери да ли се на путањи возила налази нека значајнија група облака података.

5. Закључак

Решење представљено у оквиру овог рада излаже једно решење процене раздаљине до возила, које се креће непосредно испред, у истој возној траци. Решење се базира на употреби фузије података добијених са два различита типа сензора, камере и *LIDAR*-а, а у сврху имплементације *CAS* система. Решење приказује један од начина решавања овог комплексног проблема са којим се сусреће аутомобилска индустрија, али и уводи неке од важних концепата пројектовања једног софтвера са применом у аутомобилској индустрији. Решење је тестирано у симулираном окружењу, у неколицини ситуација, које се јављају током вожње у градским условима. Решење даје добре резултате у већини ситуација, али постоје и извесна ограничења.

Решење је неопходно у неком наредном периоду унапредити и довести на неки виши ниво. Овим радом нису покривене неке од битних ситуација у којим аутомобил може да се нађе. Ради једноставности, узета је ситуација да је возна трака увек праволинијског облика, и да се за тражени објекат узима онај који се налази директно испред возила. Међутим то није случај у ситуацијама где је возна трака криволинијског облика. Као евентуално решење овог проблема представља интеракција модула за детекцију објеката и модула за детекцију возне траке. Други битан фактор који утиче на прецизност детекције јесте временска синхронизација оквира, који нам долазе са камере и *LIDAR*-а. У оквиру овог рада није подржана временска синхронизација, коју је неопходно додати у оквиру даљег развоја овог решења. Као последња ставка којом је потребно проширити постојеће решење представља смањивање зависности успешности детекције и процене удаљености препреке од *2D* детекције објекат. Као што је у поглављу “Поступак испитивања и резултати” поменуто, алгоритам је стриктно зависан од успешности детекције објекта са камере. Уколико алгоритам који служи за *2D* детекцију објеката са камере није у ситуацији да успешно

детектује објекат, услед разних спољашњих прилика, алгоритам неће пријавити постојање опасности, чак и да заиста постоји препрека испред. Услед тога је неопходно дати већу аутономију обради података са *LIDAR*-а, тако да и уколико *2D* детекција није дала информацију о постојању препреке, подаци са *LIDAR*-а буду обрађени и дају назнаку постојања препреке, чак и по цену да не знамо који тип препреке је у питању.

Овим радом је представљен један другачији приступ решавању једног од значајних изазова са којим се сусрећу аутономна возила, а то је ефикасна процена раздаљине до препреке, која се налази непосредно испред возила, а у циљу имплементације система за спречавање чеоног судара. Рад описује неке од приступа решавању овог проблема, али такође истиче и извесна ограничења, на које треба обратити пажњу и наћи адекватан приступ за њихово решавање у будућности.

6. Литература

[1] „Automated Vehicles for Safety“, https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety [приступљено: април 2020.].

[2] „SAE Internacional“, https://www.sae.org/ [приступљено: април 2020.].

[3] „ISO“, https://www.iso.org/standards.html [приступљено: април 2020.].

[4] „AUTOSAR Classic Platform“, https://www.autosar.org/standards/classic-platform/ [приступљено: април 2020.].

[5] „AUTOSAR Adaptive Platform“, https://www.autosar.org/standards/adaptive-platform/ [приступљено: април 2020.].

[6] S Chen, J Hu, Y. Shi, Y. Peng, „Vehicle-to-Everything (v2x) Services Supported by LTE-Based Systems and 5G“, 2017 IEEE Communications Standards Magazine.

[7] „ROS“, https://www.ros.org/ [приступљено: мај 2020.].

[8] „AUTOWARE foundation“, https://www.autoware.org/ [приступљено: мај 2020.].

[9] „CARLA“ симулатор, https://carla.org/ [присутпљено: мај 2020.].

[10] M. Vranjes, „Дигитална обрада слике и видеа за примену у аутономним возилима “, Универзитет у Новом Саду, Факултет Техничких Наука.

[11] „Clustering Algorithms“, https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68/ [приступљено: јун 2020.]

[12] „Kalman filter“, https://towardsdatascience.com/an-intro-to-kalman-filters-for-autonomous-vehicles-f43dd2e2004b/ [приступљено: јун 2020.].

[13] „Camera-Lidar Projection: Navigation between 2D and 3D“, https://medium.com/swlh/camera-lidar-projection-navigating-between-2d-and-3d-911c78167a94 [приступљено: мај 2020.].

[14] „YOLO“, https://pjreddie.com/darknet/yolo/ [приступљено: мај 2020.].

[15] J. Redmon, A. Farhadi, „YOLOv3: An Incremental Improvement “, 2018

[16] „AUTOWARE Camera-LIDAR Calibration Package“, https://autoware.readthedocs.io/en/feature-documentation_rtd/DevelopersGuide/PackagesAPI/sensing/autoware_camera_lidar_calibrator.html [приступљено: мај 2020.].