Сава Недељковић [1]

Резиме

У оквиру овог мастер рада развијен је сопствени алгоритам за детекцију и идентификацију ликова објеката на слици, ради препознавања технолошких ентитета у оквиру лабораторијског модела технолошког окружења. Детекција ликова објеката је базирана на примени савремених техника, у домену машинског гледања, у које се убрајају: Гаусов филтер, „Вододелница“ (енгл. „Watershed“) алгоритам, методе сегментације базиране на детекцији ивица, RGB и HSV анализа слике. Након компаративне анализе доступних метода сегментације, уочава се да је Собелова метода са најмањим шумом успела да детектује круцијалне ивице ликова објеката. Применом RGB прага извршено је раздвајање региона слике који припадају објектима од региона слике који припадају окружењу, након чега је применом HSV прага елиминисан преостали шум. Сепарација преклапајућих објеката, на сегментисаној слици, извршена је коришћењем, „Вододелница“ алгоритма, при чему је развијен и алгоритам на основу кога је извршена његова корекција. За идентификацију детектованих ликова објеката коришћене су вештачке неуронске мреже базиране на „дубоком“ машинском учењу. Како би се пронашло оптимално решење за идентификацију технолошких ентитета, разматрано је више модела конволуционих неуронских мрежа, које су процесом „дообучавања“ модификоване да одговарају постављеном проблему. Од разматраних модела најбоље резултате, у погледу тачне идентификације и брзине извршавања, показао је „дообучен“ модел AlexNet-а. Тестирање и верификација развијеног алгоритма извршена је експерименталним путем, коришћењем интелигентног мобилног робота RAICO (Robot with Artificial Intelligence based COgnition) са две Basler acA1920-25uc камере. Приликом тестирања прикупљено је укупно 400 слика при чему је алгоритам адекватно успео да идентификује приказане технолошке ентитете у 93.25% случајева. Алгоритам на основу кога се извршава детекција и идентификација ликова објеката на слици, као и алгоритам за „дообучавање“ вештачких неуронских мрежа, је развијен коришћењем MATLAB софтверског пакета.

***Кључне речи**: Методе сегментације, идентификација технолошких ентитета, вештачке неуронске мреже, „дубоко“ машинско учење, интелигентни технолошки системи, интелигентни мобилни робот*

[1] Сава Недељковић, 1152/17 – Студент друге године мастер академских студија, Универзитет у Београд, Катедра за производно машинство

САДРЖАЈ

1. Увод 9
1.1. Визуелни системи препознавања 9
1.2. Примена идентификације објеката у роботским системима 10
1.3. Интелигентни мобилни роботи 11
1.3.1 RAICO (*Robot with Artificial Intelligence based COgnition*) 11
1.3.2 Basler acA1920-25uc камера 12
2. Вештачка интелигенција 14
2.1. Вештачке неуронске мреже 15
2.2. Конволуционе неуронске мреже 21
3. Сегментација слике 27
3.1. Сегментација ликова објеката преко контура 27
3.2. Методе базиране на детекцији ивица 29
3.3. Компаративна анализа метода сегментације 31
3.4. Примена и значај Гаусовог филтера у методама сегментације слике 35
3.5. Анализа различитих метода репрезентације слике у боји и њихова примена у сегментацији слике 36
3.6. Сегментација применом „Вододелница“ алгоритма 38
4. Модул за детекцију ликова објеката применом метода сегментације слике 43
5. Идентификација детектованих ликова објеката 49
5.1. „Дообучавање“ готових модела конволуционих неуронских мрежа 49
5.2. *AlexNet* 56
6. Модул за идентификацију детектованих ликова објеката 57
7. Експериментална верификација развијеног алгоритма 61
7.1. Примена развијеног модула на реалним сликама лабораторијског модела технолошког окружења 61
7.2. Експеримент 66
8. Закључак 72
9. Литература 73
10. Прилог *MATLAB* програми 75
10.1 Главни програм за идентификацију технилошких ентитета на слици 75
10.2 Подпрограм за креирање оквира региона 82

10.3. Програм за прикупљање обучавајућих парова 83
10.4. Програм за „дообучавање“ готових модела CNN-а 91
10.5. Програм за компарацију постојећих метода сегментације 92
10.6. Програм за аквизицију слика са леве и десне камере 93

Списак табела

Табела 1 – Спефицикација Basler acA1920-25uc камере 13
Табела 2 – Спецификације Fujinon Lens DF6HA-1B сочива 13
Табела 3 – Математичка фурмулација улазних оператера [1] 16
Табела 4 – Проналажење оптималних параметара тренирања 51
Табела 5 – Резултати „дообучања“ истрениираних модела CNN-а 52
Табела 6 – Вектори дистрибуције вероватноће потенцијалних класа 59

Списак слика

Слика 1.1 – Матрица осветљености сиве боје код монохроматских слика [5] 9
Слика 1.2 – 3D модел интелигентног мобилног робота RAICO 12
Слика 1.3 – Физички приказ интелигентног мобилног робота RAICO 12
Слика 1.4 – Basler acA1920-25uc камера са компатабилним сочивом [31] 13
Слика 2.1 – Структура неуронске мреже [1] 15
Слика 2.2 – Приказ процесирајућег елемента – неурона [1] 16
Слика 2.3 – Бинарна активациона функција (лево); Сигмоидна активациона функција (десно) 18
Слика 2.4 – Архитектура конволуционих неуронских мрежа [8] 21
Слика 2.5 – Примена процеса конволуције на бинарну слику 22
Слика 2.6 – Пролаз 3D филтера кроз RGB слику [8] 23
Слика 2.7 – Сажимање просеком (лево); Сажимање максимумом (десно) 24
Слика 2.8 – ReLU активациона функција (лево); Примена ReLU активационе функције на конволуциону матрицу (десно) 24
Слика 2.9 – Архитектура мреже потпуно повезаних слојева (лево); Архитектура мреже изостављајућих слојева (десно) [17] 25
Слика 2.10 – Слојеви на излазу из CNN-а 26
Слика 3.1 – Пример сегментисане слике [34] 27
Слика 3.2 – Подела метода сегментације [12] 28
Слика 3.3 – Поређење идеалне и реалне ивице [13] 29
Слика 3.4 – Функција интензитета осветљености [3] 30
Слика 3.5 – Проналажење ивица машине 1 31
Слика 3.6 – Проналажење ивица машине 2 32
Слика 3.7 – Проналажење ивица машине 3 32
Слика 3.8 – Проналажење ивица машине 4 33
Слика 3.9 – Приказ са и без „стањивања“ детектованих линија 33
Слика 3.10 – Гаусова расподела 35
Слика 3.11 – Слика пре (лево) и након (десно) примене Гаусовог фитлера 35
Слика 3.12 – RGB модел репрезентације слике у боји [15] 36
Слика 3.13 – Слике пре и након примене RGB прага 36
Слика 3.14 – HSV модел репрезентације слике у боји [15] 37
Слика 3.15 – Слика пре и након примене HSV прага 37
Слика 3.16 – Приказ сепарације „долина“ помоћи „Вододелница“ линије [30] 38

Слика 3.17 – Слика добијена са камере (лево); Бинарна слика добијена процесирањем .38
Слика 3.18 – Приказ локалних минимума бинарне слике 39
Слика 3.19 – Мапа удаљености пиксела 39
Слика 3.20 – Резултат примене „Вододелница“ алгоритма 40
Слика 3.21 – Сепарација објеката применом „Вододелница“ алгоритма 40
Слика 3.22 – Приказ пре и након извршавања корекције „Вододелница“ алгоритма 40
Слика 3.23 – Пример „Вододелница“ алгоритма бр.1 41
Слика 3.24 – Пример „Вододелница“ алгоритма бр.2 41
Слика 3.25 – Пример „Вододелница“ алгоритма бр.3 42
Слика 3.26 – Пример „Вододелница“ алгоритма бр.4 42
Слика 4.1 – Алгоритам за детекцију ликова објеката на слици 43
Слика 4.2 – Слике добијене аквизицијом података са леве и десне камере 44
Слика 4.3 – Слике добијене проласком Гаусовог филтера 44
Слика 4.4 – Детекција ивица коришћењем Собелове методе 45
Слика 4.5 – Елиминисање шума 45
Слика 4.6 – Конектовање пиксела са оквиром слике 46
Слика 4.7 – Груписање пиксела у регионе 46
Слика 4.8 – Формирање ликова објеката применом RGB прага 47
Слика 4.9 – Елиминисање ивица зидова применом HSV прага 47
Слика 4.10 – Сепарација преклапајућих објеката применом „Вододелница“ алгоритма ..48
Слика 4.11 – Коначан облик сегментисане слике 48
Слика 5.1 – Шема „дообучавања“ мреже (*Transfer Learning*) [20] 49
Слика 5.2 – Пример „сличних“ обучавајућих парова машине 2 и машине 3 52
Слика 5.3 – Приказ тока процеса „дообучавања“ 53
Слика 5.4 – Резултати процеса „дообучавања“ кроз епохе 54
Слика 5.5 – Пример за израчунавање функције циља 55
Слика 5.6 – Матрица одлучивања 55
Слика 5.7 – Упрошћен приказ архитектуре *AlexNet*-а [17] 56
Слика 5.8 – Списак слојева унутар архитектуре *AlexNet*-а 56
Слика 6.1 – Алгоритам за идентификацију технолошких ентитета 57
Слика 6.2 – Слике добијене аквизицијом података са леве и десне камере 58
Слика 6.3 – Резултати сегментације учитаних слика 58
Слика 6.4 – Појединачне слике детекованих ликоба објеката са леве и десне камере.....59
Слика 6.5 – Резултати идентификације технолошких ентитета 60

Слика 7.1 – Примери тачне идентификације технолошких ентитета, 1 61
Слика 7.2 – Примери тачне идентификације технолошких ентитета, 2 62
Слика 7.3 – Примери тачне идентификације технолошких ентитета, 3 63
Слика 7.4 – Случај нетачно идентификованих технолошких ентитета, пример 1 63
Слика 7.5 – Случај нетачно идентификованих технолошких ентитета, пример 2 64
Слика 7.6 – Случај нетачно идентификованих технолошких ентитета, пример 3 64
Слика 7.7 – Случај нетачно идентификованих технолошких ентитета, пример 4 65
Слика 7.8 – Детектоване машине са леве и десне камере, корак 1 66
Слика 7.9 – Правилно упарени карактеристични објекти на левој камери, корак 1 66
Слика 7.10 – Правилно упарени карактеристични објекти на десној камери, корак 1 66
Слика 7.11 – Детектоване машине са леве и десне камере, корак 2 67
Слика 7.12 – Правилно упарени карактеристични објекти на левој камери, корак 2 67
Слика 7.13 – Правилно упарени карактеристични објекти на десној камери, корак 2 67
Слика 7.14 – Детектоване машине са леве и десне камере, корак 3 67
Слика 7.15 – Правилно упарени карактеристични објекти на левој камери, корак 3 67
Слика 7.16 – Правилно упарени карактеристични објекти на десној камери, корак 3 67
Слика 7.17 – Детектоване машине са леве и десне камере, корак 4 68
Слика 7.18 – Правилно упарени карактеристични објекти на левој камери, корак 4 68
Слика 7.19 – Правилно упарени карактеристични објекти на десној камери, корак 4.....68
Слика 7.20 – Детектоване машине са леве и десне камере, корак 5 68
Слика 7.21– Правилно упарени карактеристични објекти на левој камери, корак 5 68
Слика 7.22 – Правилно упарени карактеристични објекти на десној камери, корак 5.....68
Слика 7.23 – Детектоване машине са леве и десне камере, корак 6 69
Слика 7.24 – Правилно упарени карактеристични објекти на левој и десној камери, корак 6.....69
Слика 7.25 – Детектоване машине са леве и десне камере, корак 7 69
Слика 7.26 – Правилно упарени карактеристични објекти на левој камери, корак 7 69
Слика 7.27 – Правилно упарени карактеристични објекти на десној камери, корак 7.....69
Слика 7.28 – Детектоване машине са леве и десне камере, корак 8 69
Слика 7.29 – Правилно упарени карактеристични објекти на левој камери, корак 8 70
Слика 7.30 – Правилно упарени карактеристични објекти на десној камери, корак 8 70
Слика 7.31 – Детектоване машине са леве и десне камере, корак 9 70

Слика 7.32 – Правилно упарени карактеристични објекти на левој камери, корак 9 70
Слика 7.33 – Правилно упарени карактеристични објекти на десној камери, корак 9 70
Слика 7.34 – Детектоване машине са леве и десне камере.....71
Слика 7.35 – Правилно упарени карактеристични објекти на левој и десној камери.....71

Номенклатура

ВСП – **В**изуелни **с**истеми **п**репознавања

RAICO – енгл. ***R****obot with* ***A****rtificial* ***I****ntelligence based* ***Co****gnition*

PWM – енгл. ***P****ulse* ***W****idth* ***M****odulation*

DC – енгл. ***D****irect* ***C****urrent*

AV – енгл. ***A****utonomic* ***V****ehicles*

ВНМ – **В**ештачке **н**еуронске **м**реже

BPNM – „***B****ack****p****ropagation*“ **н**еуронске **м**реже

DARPA – енгл. ***D****efense* ***A****dvanced* ***R****esearch* ***P****rojects* ***A****gency*

CNN – енгл. ***C****onvolutional* ***N****eural* ***N****etwork*

RIA – енгл. ***R****obotic* ***I****ndustries* ***A****ssociation*

ISO – енгл. *International Organization for Standardization*

SGDM – енгл. ***S****tochastic* ***G****radient* ***D****escent with* ***M****omentum*

ReLU – енгл. ***Re****ctified* ***L****inear* ***U****nit*

LoG – енгл. ***L****aplacian* ***O****f* ***G****aussian*

RGB – енгл. ***R****ed* ***G****reen* ***B****lue*

HSV – енгл. ***H****ue* ***S****aturation* ***V****alue*

1. Увод

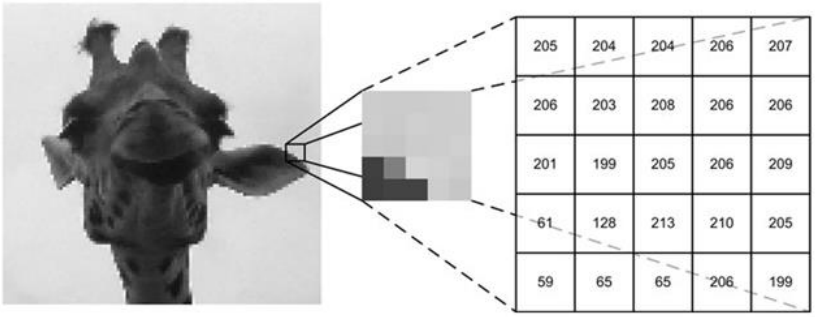
1.1 Визуелни системи препознавања (ВСП)

Визуелни системи препознавања који се користе код индустријских робота, и који су базирани на коришћењу камере као сензора, треба да буду у стању да најпре детектују, а затим и идентификују, индустријске производе комплексних геометријских облика. Извршавање поменутих акција подразумева решавање проблема везаних за одређивање позиције и оријентације објеката у сцени, у релативно кратком временском року, како би могла да се предузме правовремена акција енд-ефектора.

Основни циљ укључивања камере у систем индустријских робота јесте давање истом способност „машинског гледања“ (енгл. *Machine Vision*), како би могао да „осети” промене, настале у реалном окружењу, и да на основу стечених информација предузме одговарајуће акције задате управљачким алгоритмом. Имплементација „машинског гледања“ обухвата проблеме процесирања слике, анализе сцене, разумевања слике, као и проблеме везане за коришћење техника вештачке интелигенције у циљу генерализације стеченог знања и доношење адекватне правовремене одлуке.

Процесирање слике представља трансформацију слике из континуалног поља аналогне слике, добијене од камере, у еквивалентни дигитални облик. Под дигиталном сликом се подразумева матрица бројева чије вредности представљају ниво осветљености пиксела слике. Сваки пиксел се описује његовом позицијом на слици, величином поља и интензитетом осветљености.

Уколико се ради о монохроматским сликама (слика 1.1), слика се представља матрицом чије вредности описују интензитет осветљености пиксела, при чему се са вредношћу 0 описује апсолутно црна боја, а са вредношћу 255 апсолутно бела боја. Број могућих нијанси сиве боје се своди на 254 због меморијског капацитета стандардних PC рачунарских платформи.



Слика 1.1 – Матрица осветљености сиве боје код монохроматских слика [5]

Са друге стране, слике у боји (RGB слике) се репрезентују помоћу три матрице, при чему свака од тих матрица одговара нивоима осветљености црвене, зелене и плаве боје. Променом количине осветљеноси поменутих матрица постиже се целокупан спектар боја. Пошто рад са сликама у боји захтева веће рачунарске ресурсе од монохроматских слика, уобичајно је да се добијене слике у боји трансформишу у монохроматски облик пре него што се ступи у њихово анализирање.

Анализа сцене полази од претходно формиране дигиталне слике, са циљем да се добије жељени опис слике, који у индустријској примени првенствено подразумева детекцију објеката у сцени, односно одређивање њихове тренутне позиције и оријентације. Како би се олакшала и убрзала анализа слике, развијене су различите методе базиране на процесу симплификације комплексног објекта које подразумевају груписање идентификованих региона у скупове по одговарајућим својствима (нпр. нивоу осветљености). Овај процес се, другим речима, назива сегментација слике.

1.2 Примена идентификације објеката у роботским системима

Да би роботски системи имали способност машинског гледања, њихови визуелни системи морају не само да детектују објекте, већ и да их на адекватан начин идентификују. Под идентификацијом објеката се подразумева додељивање класе детектованим објектима на слици.

Циљ идентификације јесте довести систем на висок степен разумевања детектованих ликова, а тиме и окружења, омогућавајући лакше доношење одлука о предстојећим акцијама. Из тог разлога своју примену проналазе у бројим областима, нпр.: аутономна возила (AV), праћење објеката на видео снимку, препознавање лица, индустријска контрола квалитета, медицинска дијагностика итд.

Да би ови системи могли да се примењују, тачност идентификације детектованих ликова мора да буде на изузетно високом нивоу. У супротном може доћи до недопустиве грешке, као на пример: губљење праћеног објекта на видео снимку, возило погрешно идентификује пешака, погрешна дијагноза болести итд. Готово све методе за идентификацију објеката се базирају на машинском или „дубоком“ машинском учењу. Разлика је у томе што се код машинског учења карактеристике (енгл. *Features*) које желимо да идентификујемо морају тачно дефинисати, док се код „дубоког“ машинског учења самостално проналазе одговарајућим алгоритмом.

Методе базиране на приступу машинског учења [6]:

- *„Viola - Jonas object detection”* алгоритам,
- SIFT (*Scale-invariant feature transform*) алгоритам,
- HOF (*Histogram of oriented gradients*) алгоритам итд.

Методе базиране на приступу „дубоког“ машинског учења [6]:

- CNN (R-CNN, Fast R-CNN, Faster R-CNN),
- SSD (*Single Shot multibox Detector*) алгоритам,
- YOLO (*You Only Look Once*) алгоритам итд.

1.3 Интелигентни мобилни роботи

Први комерцијални индустријски роботи су се појавили крајем 60тих година 20. века. У то доба, они су били стационарни и извршавали су једноставне задатке манипулације и тачкастог заваривања. Постоје више начина на основу којих се могу дефинисати индустријски роботи, при чему су најприхваћенији приступи RIA и ISO:

RIA: Индустријски робот је репрограмабилни, вишефункционални манипулатор пројектован да помера материјал, делове, алате и специјалне уређаје кроз различита програмирана кретања при извршавању постављених технолошких задатака [25].

ISO: Индустријски робот је вишенаменска, репрограмабилна, манипулациона и аутоматски управљана машина, са више степени слободе кретања, која може бити у фиксном положају или мобилна, а користи се за аутоматизоване примене у индустрији [25].

Интензиван раст захтева на тржишту проузроковао је и развој индуструјских робота, тако да се уз реч робот данас често појављују атрибути интелигентан и мобилан.

Под мобилним роботом се подразумевају сви роботи који могу слободно да се крећу кроз технолошко окружење, односно који поседују могућност локомоције. Предност мобилних робота, у односу на стационарне роботе, јесте што им је радни простор значајно проширен. Постоје различита окружења у којима се мобилни роботи могу кретати, при чему разликујемо:

- Роботе који се крећу по чврстим подлогама,
- Роботе који се крећу кроз водену средину,
- Роботе који се крећу кроз ваздушну средину – дронови и
- Роботи који се могу кретати кроз више различитих окружења – амфибијски роботи.

Поред мобилности самог индустријског робота, потребно је да исти поседује висок степен интелигентног понашања како би могао да се примењује приликом извршавања савремених задатака унутрашњег транспорта. Под интелигентним роботима се подразумевају роботи који поседују могућност „разумевања” свог окружења како би били у стању да аутономно доносе правовремене одлуке о својим акцијама. Аквизиција информација о променама у окружењу извршава се помоћу сензора (камера), који су уграђени на мобилном роботу.

1.3.1 RAICO (Robot with Artificial Intelligence based COgnition)

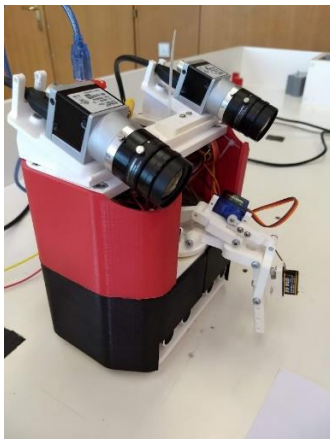
RAICO *(**R**obot with **A**rtificial **I**ntelligence based **CO**gnition)* представља један интелигентни мобилни робот, сопственог развоја, пројектован са циљем да се у оквиру постојећег лабораторијског модела технолошког окружења оствари унутрашњи транспорт и манипулација материјала.

Садржи четири основна подсистема, и то:

1. Аквизиционо – управљачки систем,
2. Погонски ситем,
3. Систем манипулације и
4. Стерео визуелни систем.

Основну компоненту аквизиционо – управљачког система чине два микроконтролера *Arduino Mega 2560 Rev3* и *Arduino Nano V3.0.* Микроконтролери су повезани са осталим подсистемима при чему се сигнал за покретање мотора шаље преко рачунара. Кретање робота остварује се помоћу два DC серво мотора *JGY371*, на које су постављена два точка обложена гумом. Управљање брзинама мотора извршава се импулсно – ширинском модулацијом (PWM). Систем манипулације је постављен са предње стране робота. Манипулација објеката, приликом опслуживања и транспорта материјала између појединих машина унутар постојећег лабораторијског модела технолошког окружења, извршава се помоћу хватача на коме се налази енд-ефектор са три степена слободе кретања. Стерео визуелни систем је базиран на коришћењу две паралелно постављене *Basler acA1920-25uc* камере. Све напоменуте компоненте су уграђене на носеће плоче, које су заједно са оплатом, израђене технологијом 3D штампе. На слици 1.2 , представљен је 3D модел интелигентног мобилног робота RAICO, док је на слици 1.3 представљен његов физички приказ.

Слика 1.2 – 3D модел интелигентног мобилног робота RAICO



Слика 1.3 – Физички приказ интелигентног мобилног робота RAICO

1.3.2 Basler acA1920-25uc камера

Како би се остварило стерео визуелно управљање на интелигентни мобилни робот RAICO уграђене су две Basler acA1920-25uc камере. Камере су паралелно постављене под углом који омогућава најбоље видно поље технолошког окружења. Као што се може и видети са слике 1.3, камере су заједно са њиховим носачима постављене на горњој носећој плочи.

Имплементацијом одговарајућег софтвера (*Image Acquisition Toolbox Support Package for GigE Vision Hardware*) омогућава се аквизиција слика помоћу MATLAB-а. Пре него што се камере могу користити, потребно је подесити жељени ниво осветљености слике (енгл. *White Balance*), као и фокус. Подешавањем нивоа осветљености се постиже изједначавање нијанси боја на слици и у стварности. Пошто камера нема аутоматско подешавање фокуса, веома је битно подесити фокус тако да се објекти, који се налазе на жељеном растојању, прикажу са што мање шума.

Основне спецификације камере дате су табелом 1.

Произвођач сензора	ON Seminiconductor
Тип Сензора	MT9P031 – CMOS
Величина сензора	4.2 x 2.4 [mm]
Максимална видео резолуција	1920x1080 px
Величина пиксела	2.2 x 2.2 [µm]
Максимални број фрејмова у секунди	26 fps
Врста комуникације	USB 3.0
Ширина, дужина и висина	29.3 x 29 x 29 [mm]
Прирубница за сочиво	C – mount
Маса камере	80g

Табела 1 – Спефицикација Basler acA1920-25uc камере

Како камере не могу радити без сочива, на основу табеле 1, потребно је одабрати компатибилна сочива. Одабрано сочиво, које се користило у реализацији овог рада јесте *Fujinon Lens DF6HA-1B*. Основне спецификације сочива дате су у табели 2, док је приказ камере, са компатабилним сочивом, дат сликом 1.4.

Примена	Камере у боји
Фокална дужина	6 [mm]
Величина отвора	F1.2-F16
Хоризонтални угао видног поља	57.3°
Вертикални угао видног поља	43,8°
Комуникација	USB 3.0
Димензије	Ø29.5 x 36.7 [mm]
Прирубница за сочиво	C – mount
Маса камере	55g

Табела 2 – Спецификације Fujinon Lens DF6HA-1B сочива

Слика 1.4 - Basler acA1920-25uc камера са компатабилним сочивом [31]

2. Вештачка интелигенција

Интелигенција или интелект (лат. *intellectus*) јесте ментална особина која се састоји од више способности: учење из искуства, адаптирање на нове ситуације, схватање и разумевање нових ситуација и коришћења стеченог знања у интеракцији са окружењем [2].

Деф: Вештачка интелигенција (машинска интелигенција) представља способност емулирања (опонашања, имитирања) или дуплирања могућности сензорског процесирања и доношења одлука унутар компјутерског система [1].

Другим речима, вештачка интелигенција представља област рачунарске науке под којом се подразумева стварање интелигентних система који поседују аутономно понашање. Због бројних предности, вештачка интелигенција налази све већу примену у готово свим људским делатностима, нпр.: фармација, медицина, маркетинг, банкарство/финансије, војна индустрија, производно машинство, авио индустрија итд.

Вештачка интелигенција се развија из дана у дан, при чему су неке од кључних области развоја [27]:

- разумевање говора,
- планирање и решавање проблема,
- самостално доношење одлука,
- истраживање терена (окружења),
- системи препознавања – анализа слике.

Учење представља процес емпиријске промене стања живог организма тако да новодобијено стање функционише боље у истоветним ситуацијама. Код машинског учења, тај процес је формализован кроз математичко-алгоритамске подлоге које треба да обезбеде интелигентном систему самостално доношење одлука приликом интеракције са окружењем.

Један начин за класификовање машинског учења је следећи:

- рутинско учење,
- учење на основу инструкција,
- дедуктивно учење,
- индуктивно учење,
- учење на основу аналогије.

Рутинско учење представља најнижи ниво машинског учења које се своди на учење напамет, тј. знање је директно уграђено у интелигентни систем. Са друге стране, учење на основу аналогије представља највиши ниво машинског учења при којем интелигентан систем има способност да самостално доноси правовремене одлуке током извршавања постављеног задатка.

Увођење вештачке интелигенције у савремене технолошке процесе представља примаран правац у развоју производних технологија у савременој индустрији. Интлигентни технолошки системи заправо обједињују компјутерски интегрисане технологије и вештачку интелигенцију. Такође, они поседују способност да самостално доносе правовремене закључке у реалном времену и то у нестационарним и динамичким срединама.

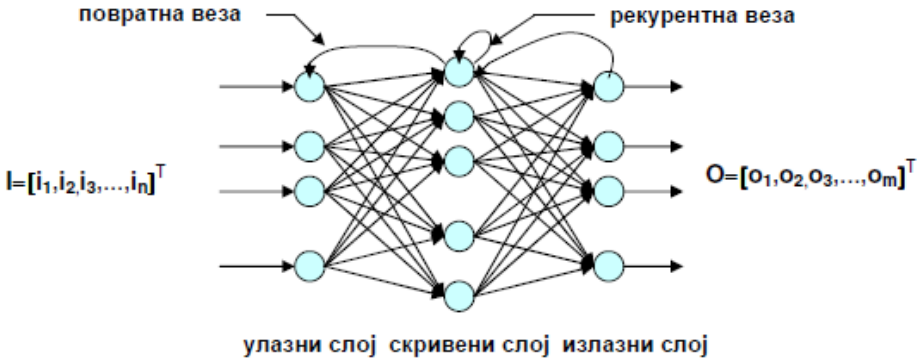
2.1 Вештачке неуронске мреже (енгл. Artificial Neural Networks)

Деф: Вештачка неуронска мрежа је парадигма вештачке интелигенције која се дефинише као конективни модел за резоновање заснован на аналогији са мозгом, уз наглашену когнитивну способност да учи и врши генерализацију стеченог знања [2].

Према DARPA (***D**efense **A**dvanced **R**esearch **P**rojects **A**gency*) пројекту у САД, неуронске мреже обезбеђују значајне предности при решавању проблема процесирања података који захтевају рад у реалном времену и интерпретацију односа између променљивих у вишедимензионалним просторима.

Погодне су за коришћење код комплексних система у машинском инжењерству (машине алатке, роботи, аутомобили, авиони итд.), као и код многих технолошких процеса, када се због непознатих и/или непотпуних правила не може успоставити општи модел.

Архитектура неуронске мреже се састоји из скупа процесирајућих елемената – неурона, међусобно повезаних везама са одговарајућим тежинским односима. Неурони су груписани у слојеве при чему по правилу мрежа поседује један улазни, један излазни и један или више скривених слојева (слика 2.1).



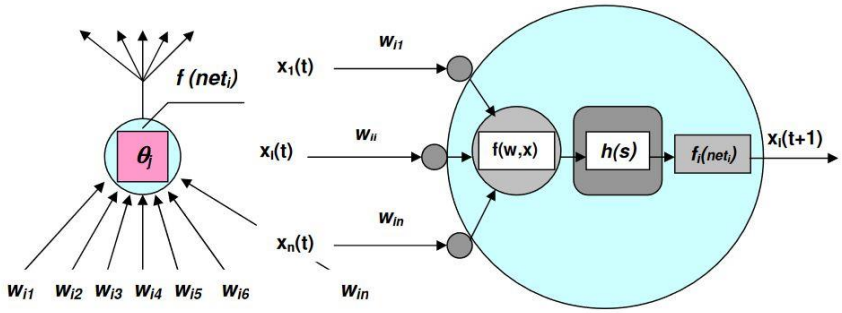
Слика 2.1 – Структура неуронске мреже [1]

Процес обучавања се остварује пресликавањем улазног у излазни простор, при чему се кроз процес минимизације грешке, применом одговарајућег алгоритма обучавања, врши модификација асоцијативне меморије, на основу које се доноси одлука о излазу из мреже. Пре него што се ступи у обучавање ВНМ, потребно је да се прикупе обучавајући парови, који представљају примере, узорке на основу којих мрежа треба да „учи“. Уобичајно је да мрежа користи 70% обучавајућих парова за тренирање, 20% за тестирање и 10% за валидацију.

Преко улазног слоја прикупљени подаци из околине се прослеђују ка скривеним слојевима где се извршава њихово процесирање, а затим се резултат репрезентује на излазном слоју. Слојеви „комуницирају“ тако што излаз сваког неурона из претходног слоја представља улаз у неуроне наредног слоја, на основу чега се закључује да неурон представља један вишепреносни систем. Јачина веза којом су неурони повезани назива се тежински фактор (енгл. *Weight*). Главна улога неурона јесте примање и обрада сигнала/информација, добијених од окружења или од другог неурона, формирајући јединствени излаз. У зависности од вредности излаза неурона, као и прага активације, везе између неурона могу бити екситационе (побуђивачке) или инхибиторне (умирујуће).

Неурон се састоји из следећа три елемента (слика 2.2):

- ***улазни оператор f(w,x)***, који обједињује улазе и тежинске односе међусобних веза *w*, формирајући јединствену вредност *s* која се прослеђује функцији преноса;
- ***функција преноса h(s)***, која врши интерполацију излаза из неуронског улазног оператора, формирајући потребну вредност за активациону функцију;
- ***активациона функција $f_i(net_i)$***, која обрађује излаз функције преноса и управља излазном вредношћу неурона.



Слика 2.2 - Приказ процесирајућег елемента – неурона [1]

Математичка формулација улазних оператера који се најчешће примењују су приказани у табели 1.

Тип улазног оператора	*Математички излаз*	*Напомене*
Линеаран	$net_j = \sum w_{ij} x_j$	*Линеарна функција*
Тежинска сума	$net_j = \sum w_{ij} x_j + \theta_j$	*Најчешће коришћен; у неким случајевима праг* $\theta_j = 0$
Повратна веза	$net_j = \alpha net_{staro} + \beta \sum w_{ij} x_j$	α, β *су тежинске константе*

Табела 3 - Математичка формулација улазних оператера [1].

Након дефинисања стимулације улаза, остварује се конвертовање улазне вредности у активациону вредност тј. долази до активације или инхибиције неурона, која је експлицитна функција улаза мреже ка неурону у временском домену, што се може приказати једначином:

$$a_i = net_i(t) \tag{1}$$

Такође, треба водити рачуна о томе да активациона функција може да зависи и од активације неурона у претходном временском тренутку, при чему се активација неурона тада приказује следећом једначином:

$$a_i = F_i(a_i(t-1), net_i(t)) \tag{2}$$

Активационе функције имају веома значајну улогу у свим моделима вештачких неуронских мрежа. Као што име сугерише, на основу ње се доноси одлука о активацији неурона. Уобичајно је да се користи иста активациона функција за све неуроне у мрежи или унутар истог слоја.

Активационе функције које се користе у ове сврхе могу бити:

- Линеарне,
- Бинарне,
- Сигмоидне,
- Компетитивне и
- Гаусове.

Линеарна активациона функција (једначина 3) представља једну од најједноставнијих активационих функција. Њена функционалност се заснива на сумирању свих улазних параметара у неурон, пропуштајући их на излаз из неурона у интегрисаном кохерентном облику.

$$f_i^l\big(net_i(t)\big) = net_i(t) \tag{3}$$

Бинарна активациона функција (једначина 4, слика 2.3 - лево) обезбеђује неурону два стабилна стања, активно и неактивно. Између ова два стања, за разлику од осталих активационих функција, постоји тачка транзиције Ω, а не праг. Вредност тачке транзиције се обично бира да буде нула јер у том случају неурон има улогу детектора поларитета. Уколико је улазна активација у неурон позитивна и већа од нуле, неурон „окида“ и генерише активни излаз, док је у свим другим случајевима излаз неактиван. Из претходно наведеног може се закључити да је у питању одскочна функција.

$$f_i^l\big(net_i(t)\big) = \begin{cases} 1 & \text{ако је } net_i(t) > \Omega \\ 0 & \text{остало} \end{cases} \tag{4}$$

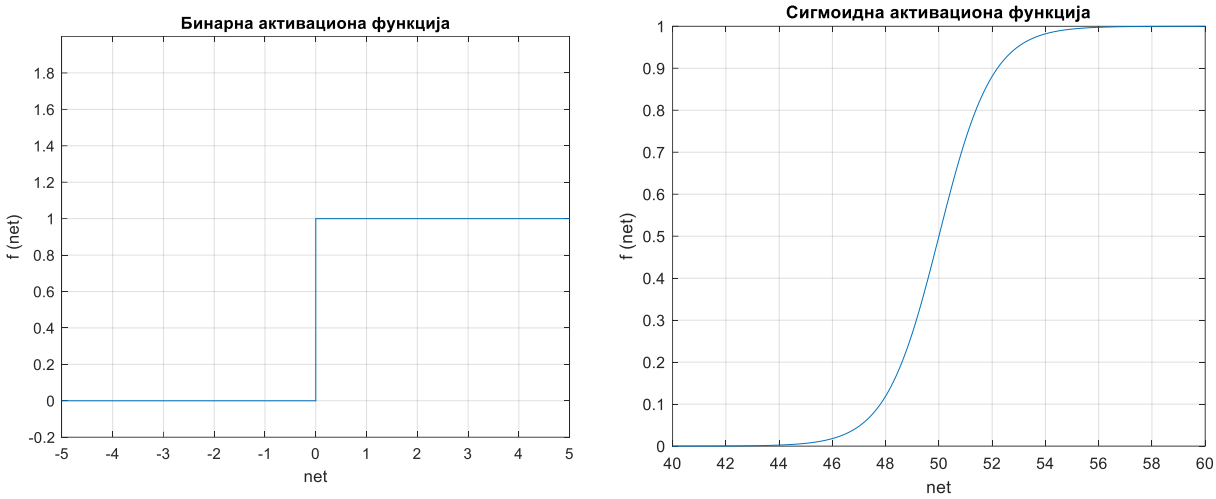
Сигмоидна активациона функција (једначина 5, слика 2.3 - десно) представља најзаступљенију активациону функцију. Као и бинарна активациона функција, она генерише излазни сигнал са два стабилна стања, с тим што код сигмоидне активационе функције постоји праг транзиције, а не тачка. Значај сигмоидне активационе функције се огледа у томе што је она математички континуална и два пута диференцијабилна. Такође, треба нагласити да је ова функција нелинеарна и неопадајућа. Најопштији математички облик ове функције јесте следећи:

$$f_i^s\big(net_i(t)\big) = \frac{1}{1+e^{-(net_i(t)-\theta)/\tau}} \tag{5}$$

Из ове једначине лако се може подесити тачка прелаза као и облик криве варирањем вредности θ и τ. Код специјалних неуронских мрежа (једначина 6), на облик криве утиче независни параметар λ, при чему је математички облик фуније следећи:

$$f_i^s\big(net_i(t)\big) = \frac{1}{2}\big(1 + \tanh(\lambda net_i(t))\big) \tag{6}$$

Уколико је излаз из неурона који користи сигмоидну активациону функцију активан, значи да је присутна специфична комбинација карактеристика улазног вектора, док неактивни излаз из неурона имплицира одсуство тих карактеристика.



Слика 2.3 - Бинарна активациона функција (лево); Сигмоидна активациона функција (десно)

Компетитивна активациона функција (једначина 7) посматра све неуроне из претходног слоја при чему само један неурон може бити „окидајући“. Неурони из претходног слоја се међусобно „такмиче“, где „победник такмичења“ узрокује активан излаз док сви остали неурони остварују неактиван излаз. Компетитивне активационе функције проналазе велику примену код конволуционих неуронских мрежа где се „такмичењем“ неурона врши класификација објеката.

$$f_i^k\big(net_i(t)\big) = \begin{cases} 1 & \text{ако је } net_i(t) = \max\{net_i(t)\}, \quad 1 \le j \le n \\ 0 & \text{остало} \end{cases} \tag{7}$$

Поред неурона, у основне компоненте мреже се убрајају још и:

- ***Топологија мреже***, којом се описује начин, врста и структура повезивања неурона у функционалну целину узимајући у обзир тежинске односе између неурона;

- ***Алгоритам обучавања***, који представља начин одређивања и модификације тежинских односа између неурона свих скривених слојева.

Алгоритми обучавања се могу класификовати у две категорије:

- Алгоритми са надгледаним процесом обучавања (супервизорско учење)
- Алгоритми са ненадгледаним процесом обучавања (несупервизорско учење)

Код супервизорског учења постоји „учитељ“ који након сваке итерације проверава напредак „ученика“ сходно уведеном критеријуму евалуације, док се код несупервизорског учења на основу скупа обучавајућих парова самостално издвајају и уче основне карактеристике улаза.

Са друге стране, алгоритми обучавања се могу класификовати у односу на поступак увођења елемената обучавајућих парова у вештачку неуронску мрежу током учења, при чему разликујемо:

- Неадаптивне алгоритме обучавања (*off-line*)
- Адаптивне алгоритме обучавања (*on-line*)

Најзначајнији алгоритми учења који се примењују су: градијенти поступак, метод најмањих квадрата, Видроу – Хофово правило, Њутн – Рафсонов метод и Левенберг – Маркеов алгоритам.

Градијентни поступак спада у најпримењивије поступке приликом одређивања оптималне вредности тежинских односа. Базира се на одређивању градијента функције циља који једнозначно дефинише правац највећег опадања грешке. Поступак се итеративно понавља све док добијена грешка не буде на жељеном нивоу. Овим поступком алгоритам успешно конвергира ка локалном минимуму, али не и ка глобалном. Тај проблем се може успешно решити адекватним одабиром параметара учења μ. Процес обучавања код градијентног поступка је спор из разлога што се тежински односи мењају при свакој итерацији. Увођењем стохастике у процес израчунавања функције циља, побољшава се избегавање локалног минимума функције циља. Како би ковергенција била бржа, евалуација функције циља се може извршавати над одређеним бројем улаза, након чега се на основу сумираних вредности функције циља врши промена тежинских односа.

Брзина конвергенције се даље може побољшати увођењем утицаја тренда тј. итерције у промену градијента. На тај начин, вредност функције циља зависи не само од тренутне вредности локалног градијента, него и од тренда који до тог тренутка постоји. Овакав приступ израчунавања функције циља се назива ***стохастички градијентни поступка са моментом*** (SGDM). На овај начин се такође спречава „заробљавање“ конвергенције у локалном минимуму и омогућавања конвергенција ка глобалном минимуму.

Метод најмањег квадрата представља поступак одређивања оптималних вредности тежинских односа тако да буде задовољена критеријумска функција дата једначином 8:

$$J(W) = \frac{1}{2}\sum_{k=1}^{n}(Y^{(k)} - net\left(X^{(k)}, W\right))^2 \qquad (8)$$

где *k* представља пример из скупа обучавајућих парова, $X^{(k)}$ одговарајући „улаз“, W вектор тежинских односа између појединих одговарајућих неурона, $net\left(X^{(k)}, W\right)$ стварни „излаз“ вештачке неуронске мреже, $Y^{(k)}$ – жељени „излаз“ вештачке неуронске мреже и $J(W)$ функцију циља.

Изједначавањем система једначина са нулом, могу се одредити вредности компонената вектора тежинских односа. Систем једначина који се на крају добије се назива систем нормалних једначина и представља најбољу оцену параметара линеарног модела на основу прикупљених мерења.

Видроу – Хофово правило представља адаптивни вид обучавања вештачких неуронских мрежа. За разлику од осталих алгоритама учења, Видроу – Хофов алгоритам тражи минимум парцијалне функције циља. Самим тим, модификација тренутних тежинских односа је пропорцијална грешци, што значи да уколико је вредност излаза из мреже блиска жељеној вредности, неће доћи до промене тежинских односа.

Њутн – Рафсонов метод се примењује у проблемима оптимизације, као и приликом одређивања тежинских односа између неурона у одговарајућим слојевима мреже. За разлику од градијентног поступка, Њутн – Рафсонов метод узима у обзир локалну закривљеност функције циља приликом одређивања минимума што резултира вишом

тачношћу решења. Међутим, пошто је за израчунавање функције циља, у којој фигуришу матрице другог реда, потребно знатно више времена у односу на градијентни поступак, Њутн – Рафсонов метод ће увек спорије конвергирати.

Левернберг – Маркеов алгоритам је први пут формулисан 1944. године, а затим унапређен 1963. године. Од тада па до дан данас налази велику примену приликом обучавања вештачких неуронских мрежа. Алгоритам полази од Њутновог поступка при чему је Левенберг предложио другачији приступ модификације тежинских односа који представља „златну средину“ између алгоритма најмањег градијента и Њутновог поступка. Наиме, пре него што се алгоритам достигне минимум грешке функције циља примењује се алгоритам најмањег градијента, а након тога се узима у обзир локална закривљеност, све у циљу брже конвергенције. Основна предност овог алгоритма је та што је избегнуто одређивање матрице другог реда, па је самим алгоритам знатно бржи.

Најпознатији модели вештачких неуронских мрежа су [1]:

- Перцептрон;
- „Backpropagation“ неуронска мрежа;
- Асоцијативне неуронске мреже;
- *Hopfield*-ове неуронске мреже;
- ART неуронске мреже (ART-1, ART-2, ART-3);
- *Fuzzy* асоцијативне неуронске мреже;
- Самоорганизујуће неуронске мреже;
- Конволуционе неуронске мреже.

Током израде овог рада, коришћена је конволуциона неуронска мрежа, тако да ће у наставку она бити детаљно елаборирана.

2.2 Конволуционе неуронске мреже (енгл. Convolutional Nerual Network)

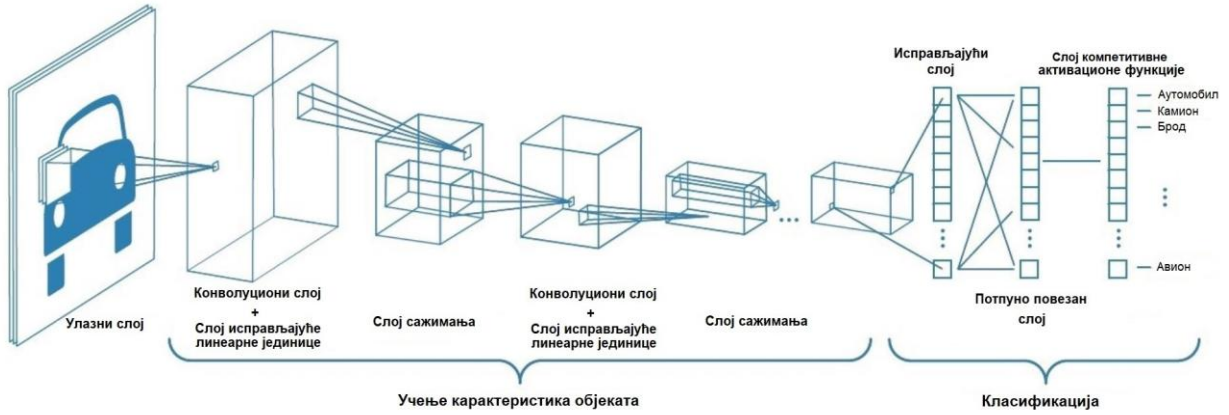
Конволуционе неуронске мреже (CNN) представљају један биолошки инспирисан модел вишеслојног перцептрона. По својој архитектури убрајају се у „дубоке“ неуронске мреже, које проналазе широку примену код визуелних система препознавања. Захваљујући смањеном броју параметара, који утичу на класификацију, као и могућности поновне употребе тежинских односа, CNN су у основи имплементације многих *state-of-the-art* решења у области машинског гледања.

CNN су пројектоване тако да буду погодне за рад са 2D структурама, као што су монохроматске слике или говорни сигнали, док због интензивног развоја у овој области постоје CNN које су погодне за рад и са 3D структурама. Како су модели CNN-а пројектоване да извршавају процесе над сликом дефинисане резолуције, све слике које се прослеђују мрежи морају имати ту резолуције. Да би се убрзао процес обучавања и примене CNN-а, уобичајно је користити слике знатно мањих резолуција.

CNN се састоје од једног улазног, једног излазног и комбинације више скривених слојева. Најчешће коришћени скривених слојеви су:

- Конволуциони слој (енгл. *Convolution layer*),
- Слој сажимања (енгл. *Pooling layer*),
- Слој исправљајуће линеарне јединице (енгл. *ReLU layer*),
- Потпуно повезан слој (енгл. *Fully Connected layer*) и
- Изостављајући слој (енгл. *DropOut layer*).

Архитектура CNN варира у зависности од задатка који мрежа треба да изврши (слика 2.4).



Слика 2.4 – Архитектура конволуционих неуронских мрежа [8]

Сваки модел CNN-а се карактерише бројем и редоследом слојева, при чему је типичан распоред операција приликом класификације објеката следећи: над учитаном сликом се прво извршава процес конволуције, у коме се детектују карактеристике објеката, након чега следи процес нормализације применом исправљајуће линеарне јединице. Затим се, проласком кроз слој сажимања смањује резолуција слике, чиме се смањује и број параметара мреже, чинећи процес обучавања бржим. Након коначног броја понављања поменутих слојева, мрежа на основу детектованих карактеристика објеката извршава класификацију коришћењем компетитивне активационе функције.

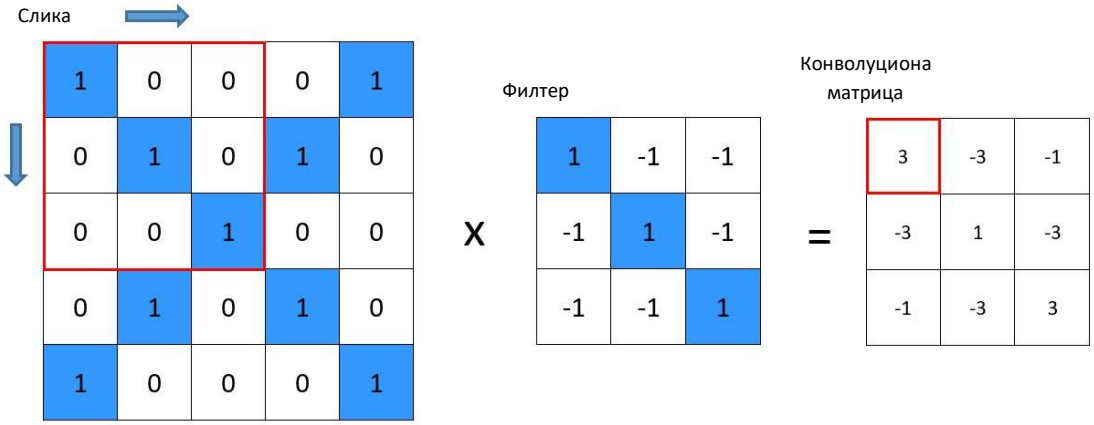
Конволуциони слој (енг. Convolution layer)

Конволуциони слој представља основни градивни блок CNN-а у којима се, као што име сугерише, обавља процес конволуције. Конволуција представља поступак примене коволуционог филтера на процесираној слици, при чему се у зависности од филтера детектују одређене карактеристике објекта, као што су на пример вертикале, хоризонталне и косе ивице. Типично је да се унутар једног конволуционог слоја примени више филтера како би се у једном пролазу детектовало више карактеристика. Све пронађене карактеристике објекта учествују у доношењу коначне одлуке о којој класи објекта се ради.

Конволуциони филтер (енгл. *Kernel*) представља генерализовани линеарни модел за регион слике на који се примењује. Другим речима, филтер представља вишедимензионалну матрицу знатно мањих димензија у поређењу са датом сликом. Типична величина дводимензионалних филтера је 2x2, 3x3 и 5x5 пиксела при чему су вредности унутар филтера реалне, и могу бити позитивне или негативне.

На слици 2.5, дат је упрошћен пример процеса конволуције. У овом примеру за дату бинарну слику, димензија 5x5 пиксела, применом конволуционог филтера величине 3x3 пиксела извршена је детекција левих косих линија. Црвеном бојом је означен део слике на који се тренутно примењује филтер. Сваки пиксел слике се множи са кореспондентном вредностима филтера, при чему се вредност конволуционе матрице формирају сумирањем добијених вредности, нпр.:

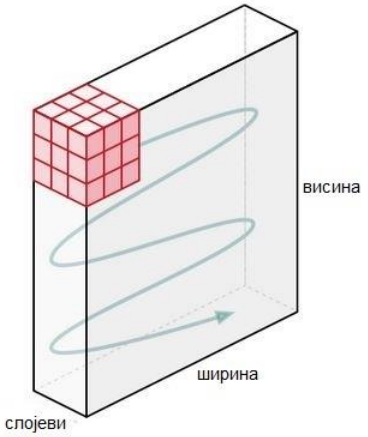
$$C(1,1) = 1 * 1 + 0 * (-1) + 0 * (-1) + 0 * (-1) + 1 * 1 + 0 * (-1) + 0 * (-1) + 0 * (-1) + 1 * 1 = 3$$



Слика 2.5 – Примена процеса конволуције на бинарну слику

Померањем филтера по хоризонтали и вертикали, за задати корак *S*, формира се коначан облик матрице конволуције *C*. У овом примеру *S = 1*, што значи да ће померање бити по један пиксел, па ће самим тим добијена матрица конволуције бити величине 3x3 пиксела. Вредности унутар конволуционе матрице указују на вероватноћу да се у кореспондентним пољима слике налази тражена карактеристика.

У општем случају овај поступак је знатно компликованији (слика 2.6). Пре свега, слике у боји представљају тродимензионалну структуру, па самим тим захтевају примену тродимензионалног филтера. Пиксели матрица филтера се множе са пикселима матрица слике, при чему се формирају три конволуционе матрице. Коначан облик матрице се формира сумирањем вредности пиксела истих позиција добијених конволуционих матрица.



Слика 2.6 – Пролаз 3D филтера кроз RGB слику [8]

Иако потпуно повезане неуронске мреже, попут „*Backpropagation*“ неуронске мреже (BPNM), могу да се користе за учење карактеристика, као и за класификацију података, CNN су се због своје једноставности и мањег броја потребног израчунавања показале као бољи класификатори. Наиме, код BPNM је потребан велики број неурона у улазном слоју којим би се репрезентовали пиксели слике. Како број тежинских односа зависи и од броја неурона у улазном слоју, процесом обучавања би било потребно да се сви они мењају, што захтева много времена. На пример, потпуно повезан слој за слику величине 100x100 пиксела има 10 000 наурона у узлазном слоју.

Коришћењем CNN-а број параметара је знатно смањен, па је самим тим процесирање слике знатно брже и ефикасније. Пиксели унутар матрице конволуције заправо представљају неуроне у конволуционом слоју који су носиоци информације за своје рецептивно поље. CNN узима у обзир просторну зависност пиксела током процеса обучавања, док се код BPNM пиксели слике представљају у векторској форми, што је још једна предност CNN-а у односну на BPNM.

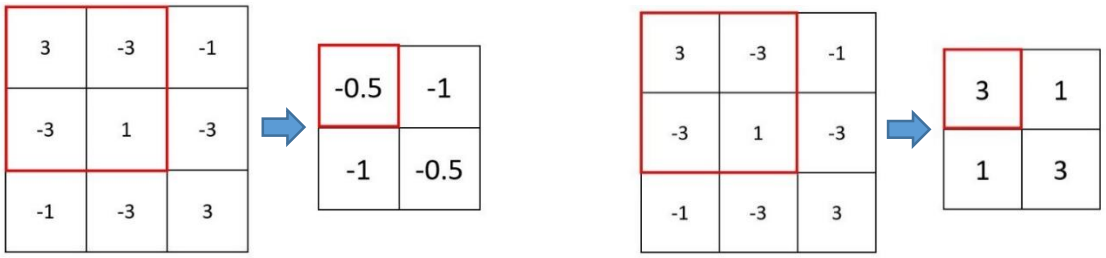
Слој сажимања (енгл. Pooling layer)

Слојеви сажимања се користе у циљу прогресивног смањења резолуције слике, а самим тим и броја параметара, што доводи до повећања брзине процесирања. Свака CNN садржи одређени број сажимајућих слојева који се јављују након процеса конволуције. Увођењем слојева сажимања у архитектуру CNN-а, знатно се смањује њено време извршавања, као и потребно време тренирања, омогућавајући њихову примену у реалном времену. Пиксели сажете слике представљају просечну или максималну вредност кореспондентних региона на слици пре сажимања.

Постоје више начина на основу којих се може извршити сажимање слике, при чему се издвајају следеће методе:

- ***Сажимање просеком* (*енгл. Average Pooling*)** - вредност неурона у слоју сажимања је једнака просечној вредности групе пиксела (слика 2.7 - лево)

- ***Сажимање максимумом* (*енгл. Max Pooling*)** - вредност неурона у слоју сажимања је једнака максималној вредности групе пиксела (слика 2.7 – десно)

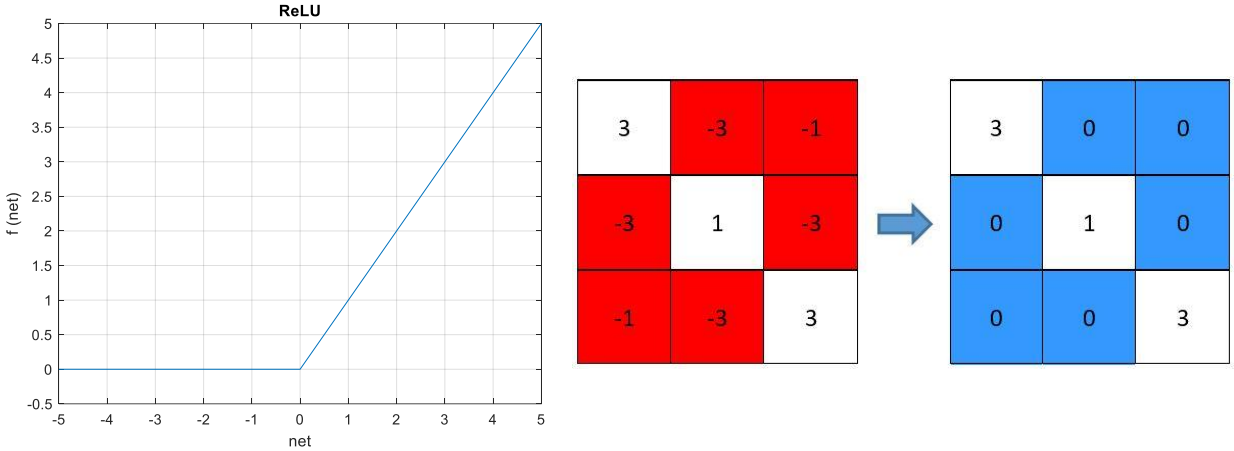


Слика 2.7 – Сажимање просеком (лево); Сажимање максимумом (десно)

Слој исправљајуће линеарне јединице (енгл. Rectified Linear Unit layer)

Слој исправљајуће линеарне јединице обично следи после конволуционог слоја како би се извршила нормализација добијене матрице. У овом слоју се применом ReLU активационе функције, чија је математичка формулација дата једначином 9., ефективно уклањају све негативне вредности у матрици тако што се претварају у нуле (слика 2.8).

$$f(net_i) = \max(0, net_i) \tag{9}$$



Слика 2.8 – ReLU активациона функција (лево); Примена ReLU активационе функције на конволуциону матрицу (десно)

Применом ове активационе функције извршава се боља апроксимација нелинеарности функције одлучивања, без нарушавања слике добијене процесом конволуције. Постоје и друге активационе функције којима се повећава апроксимација, нпр. хиперболички тангенс (једначина 10) и сигмоидна активациона функција (једначина 11). Међутим коришћењем ReLU активационе функције, тренирање неуронске мреже је и до неколико пута брже, у односу на друге активационе функције, без значајних последица на тачност генерализације, па се из тих разлога она знатно више примењује.

$$f(x) = \tanh(x)\,, f(x) = |\tanh(x)| \tag{10}$$

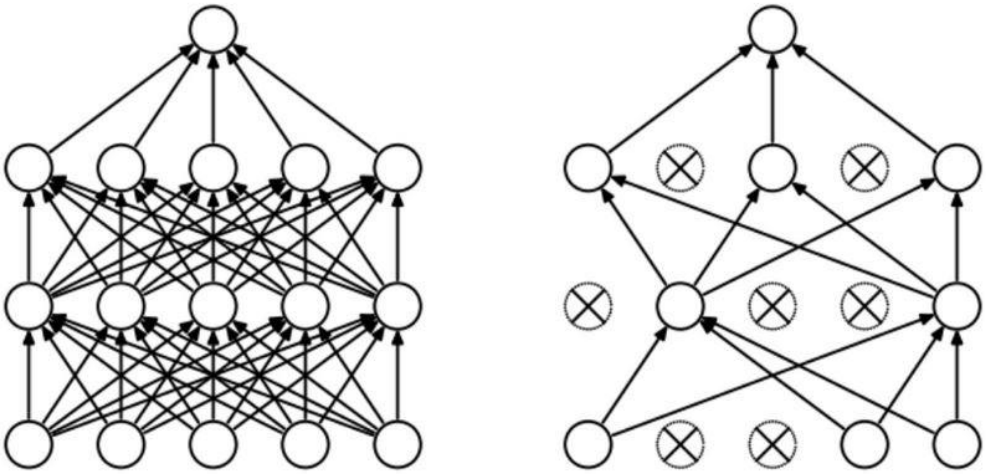
$$\sigma(x) = (1 + e^{-x})^{-1} \tag{11}$$

Изостављајући слој (енгл. DropOut layer)

Уколико сви неурони у слоју учествују у процесу учења, они почињу да развијају међусобну зависност, што значи да се индивидуални утицај сваког неурона смањује при чему лако може доћи до претренираности мреже. Један од начина да се избегне претренираност јесте увећење изостављајућег слоја [17].

У изостављајућем слоју одређени број неурона не учествује у даљем процесу учења, као и ни у промени тежинских односа. Тиме се омогућава мрежи да, са различитим неуронима, научи исте карактеристике објеката. Број неурона који се изоставља зависи од сложености архитектуре неуронске мреже, при чему се углавном изоставља 50% насумично изабраних неурона.

На слици 2.9, представљен је пример архитектуре потпуно повезане мреже пре и након изостављања пиксела. Јасно је да, због мањег броја неурона, тренирање мреже са изостављајућим слојевима постаје знатно брже, али као последица тога број итерација се практично дуплира. Код CNN-а, није карактеристично да се јавља више узастопних изостављајућих слојева већ само један и то након потпуно повезаних слојева.



Слика 2.9 – Архитектура мреже потпуно повезаних слојева (лево); Архитектура мреже изостављајућих слојева (десно) [17]

Потпуно повезан слој (енгл. Fully Connected layer)

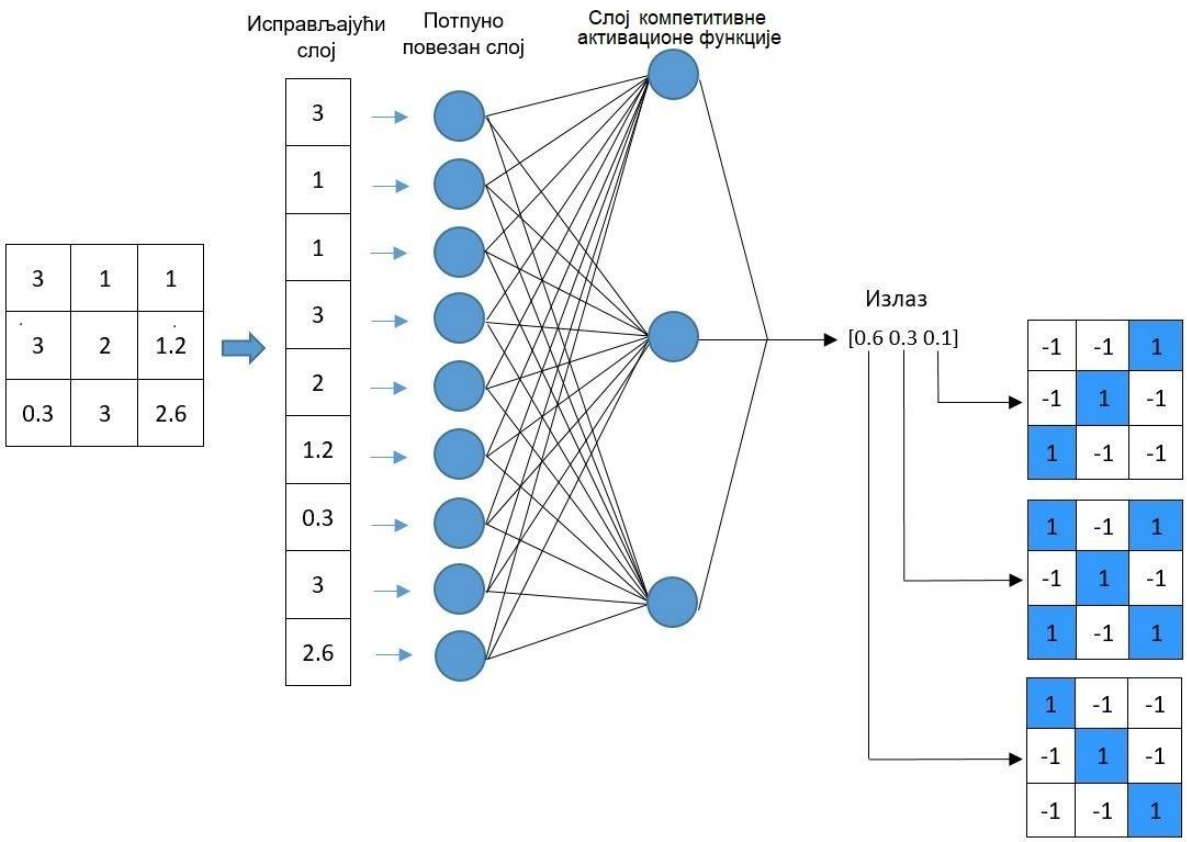
Потпуно повезани слојеви, по својој архитектури, подсећају на „*Backpropagation*“ неуронску мрежу. Сви неурони слоја су повезани са неуронима из претходног и наредног слоја. Потпуно повезани слојеви се обично постављају међу последњим слојевима мреже, чиме се постиже могућност учења нелинеарних комбинација карактеристика слике добијених проласком слике кроз претходне слојеве.

Између последњег потпуно повезаног слоја и излазног слоја налази се слој компететивне активационе функције (енгл. *Softmax layer*) чија је математичка формучација дата једначином 12. *Softmax* активациона функција претвара вредности из претходног слоја у вероватноћу. Као излаз из слоја добија се вектор који представља дистрибуцију

вероватноће потенцијалних исхода. Пошто се ради о компететивној активационој функцији, излаз из мреже се одређује на основу „победника такмичења“, односно неурона са највећом вероватноћом. Математичка формулација *Softmax* функције је следећа:

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_i}} \tag{12}$$

На слици 2.10, приказан је поступак проласка добијене матрице кроз исправљајући слој (енгл. *Flatten layer*), а затим њен даљи пролаз до излаза из мреже. Број неурона у исправљајућем слоју зависи од укупног броја чланова у добијеној конволуционој матрици, док се број неурона у потпуно повезаном слоју дефинише приликом одабира архитектуре. Ради лакше репрезентације, број неурона у исправљајућем слоју и потпуно повезаном слоју, на слици 2.10, узет да буде исти.



Слика 2.10 – Слојеви на излазу из CNN-а

3. Сегментација слике

3.1 Сегментација ликова објеката

Сегментација слике, у домену машинског гледања, представља поступак поделе дигиталне слике на регионе, односно скупове пиксела, и њихова екстракција од позадине. Другим речима, сегментација представља процес „додељивања ознаке“ сваком пикселу на слици, тако да пиксели са истом ознаком имају сличне карактеристике, као што су нпр.: интензитет осветљености, нијанса боје, текстура итд. Сликом 3.1, представљен је пример сегментисане слике, где су сви пиксели слике додељени региону пса, мачке или окружења.

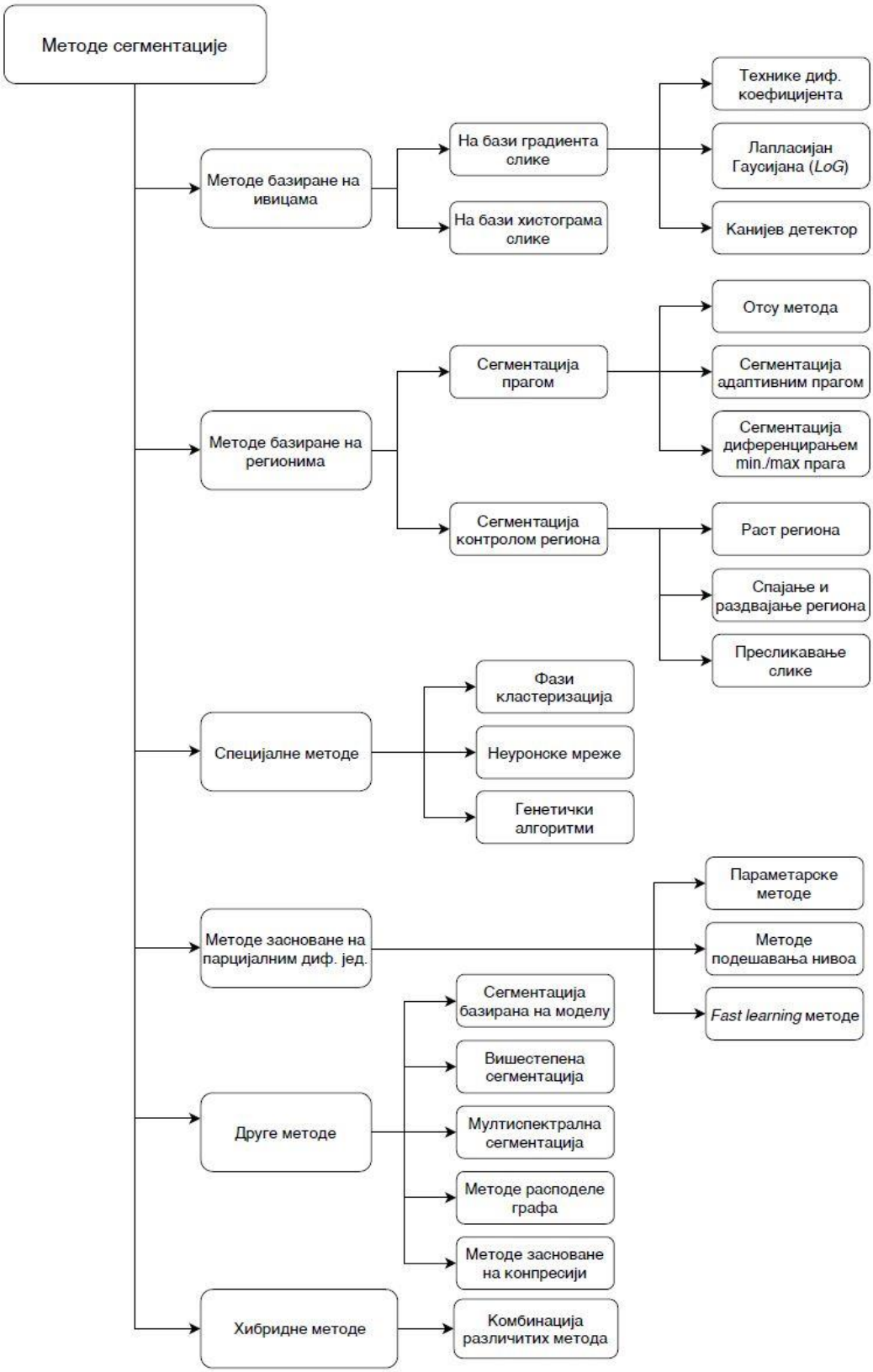
Слика 3.1 – Пример сегментисане слике [34]

Циљ сегментације јесте да се поједностави и/или промени репрезентација слике у нешто што је интуитивно и лакше за анализирање [12]. Резултати сегментације нам представљају веома значајну целину у свим наредним поступцима анализе слике, као што су репрезентација и опис објеката, мерење одређених карактеристика, праћење објеката у „online“ режиму, идентификација објеката, интерпретација сцене итд.

Овај поступак представља неизбежан корак приликом обраде слика, што подстиче развој различитих метода. Објективно посматрано, не постоје универзалне технике које се могу користити за било коју слику, тј. ниједна метода није савршена. Свака метода поседује одређене предности и недостатке у односу на друге методе, па се самим тим у зависности од проблема, који се решава, врши избор методе за сегментацију.

Постоји више приступа на основу којих можемо поделити методе сегментације, међу којима је најопширнија подела дата радом [12]. Према слици 3.2, уочава се да се у ове методе убрајају и хибридне методе, које су настале као резултат комбиновања две или више метода са циљем добијања бољих резултата сегментације слике. На основу ове поделе примећује се да, заједно са хибридном методом, постоје укупно 6 основних метода сегментације при чему свака од њих поседује одговарајуће под-категорије.

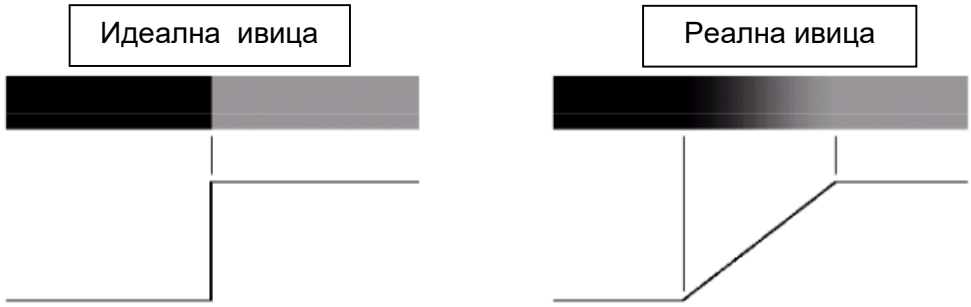
У оквиру овог рада, приликом идентификације ликова објеката на слици, разматраће се сегментација помоћу метода базираних на детекцији ивица, која ће у наставку бити детаљно елаборирана.



Слика 3.2 – Подела метода сегментације [12]

3.2 Методе сегментације базиране на детекцији ивица

Ивице представљају један скуп повезаних 2D пиксела који се налазе на граници између различитих региона, код којих постоје значајне промене интензитета осветљености. Ове методе сегментације се управо заснивају на издвајању таквих врста дисконтинуитета, при чему се сви пиксели, унутар простора ограниченог њима, сматрају да припадају истом региону, односно групацији пиксела. Идеална ивица (слика 3.3) би представљала ивицу са наглим преласком интензитета осветљености из једног пиксела у други. У пракси тај нагиб је знатно блажи због „замућења“ (енгл. *Blur*) које настаје услед карактеристика система за аквизицију.



Слика 3.3 – Поређење идеалне и реалне ивице [13]

Сегментација методама базираним на детекцији ивица свакако спада у комплексније задатке при обради слике, поготово уколико се ради о сликама у боји (RGB слике). Приликом коришћења ових метода постоје многобројни проблеми које треба решити на што ефикаснији начин, као што су нпр. [13]:

- промена осветљења на слици,
- динамичка позадина слике,
- шум, који има велики утицај на обликовање ивице,
- лажна детекција ивице.

Пре него што се ступи у издвајање ивица контура слике, неопходно је поседовати следеће информације о [1]:

- локалној оријентацији елемената ивице,
- контрасту осветљености суседних региона,
- ширини појаса елемената ивице,
- одређивању положаја тачке ивице,
- интензитету осветљености ивичних пиксела слике,
- средњој вредности интензитета осветљености суседних региона.

Пошто се слика меморише као низ дискретних пиксела, сматра се да је интензитет осветљености континуална функција унутар 2D простора слике. На слици 3.4, је дат пример дијаграма тока поменуте функције, означеном црвеном бојом, дуж једне врсте пиксела (нпр. дуж x осе). Поред функције осветљености, на дијаграму је приказана промена градијента функције, означена плавом бојом, као и промена Лапласијанове функције, означена зеленом бојом.

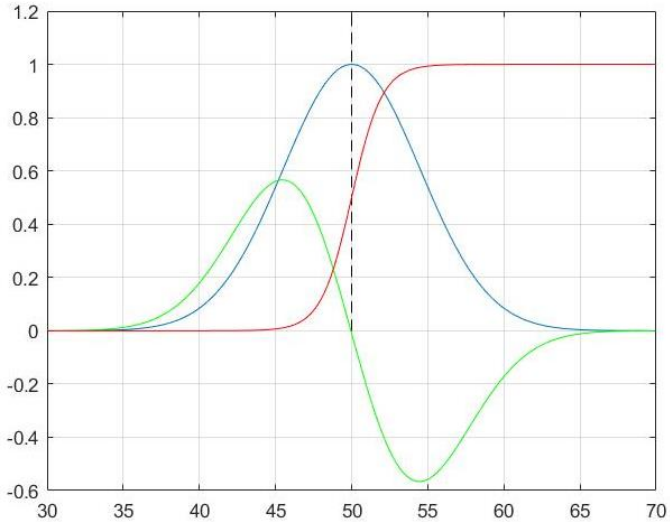
Са дијаграма се може приметити да у тачки где функција осветљености има максимални нагиб, градијент функције (једначина 13) достиже своју максималну вредност. То имплицира да је у том пикселу дошло до прелаза из једног нивоа осветљености у други, па се самим тим тај пиксел може сматрати као ивични.

$$\nabla I = \left(\frac{\partial}{\partial x} + \frac{\partial}{\partial y}\right) I = \frac{\partial I}{\partial x} + \frac{\partial I}{\partial y} \tag{13}$$

Други приступ одређивања ивичних пиксела јесте помоћу Лапласијан функције (једначина 14), која у тачки максималног нагиба функције осветљености мења знак.

$$\nabla^2 I = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{14}$$

Дакле, ивични пиксели тј. пиксели контура објеката, представљају пикселе код којих градијента функције осветљености достиже свој максимум, односно код којих Лапласијан функција врши промену знака (слика 3.4).



Слика 3.4 – Функција интензитета осветљености [3]

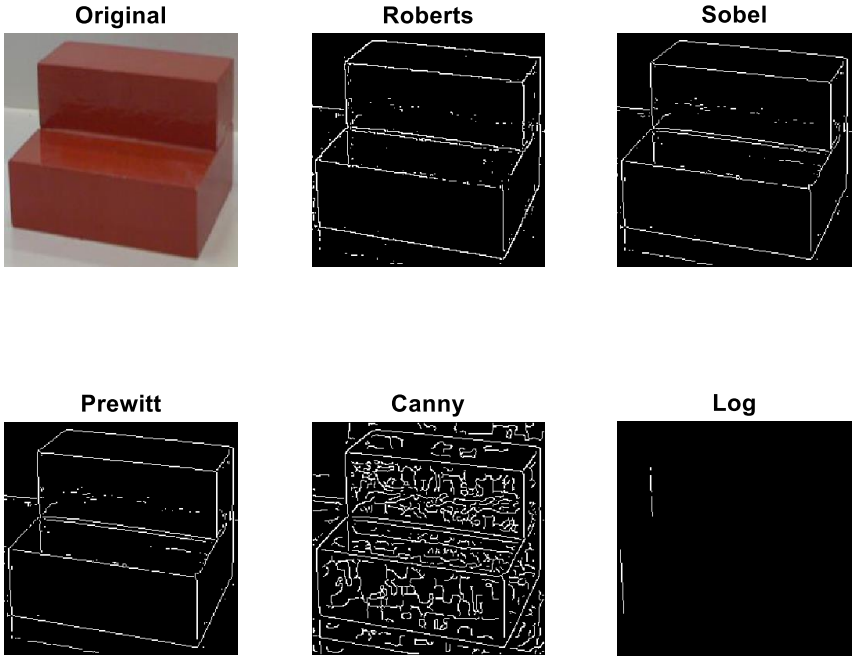
3.3 Компаративна анализа метода сегментације преко контура

У зависности од начина проналажења тачака контуре, као и начина апроксимирања функције осветљености, у методе базиране на детекцији ивица се могу сврстати следеће методе:

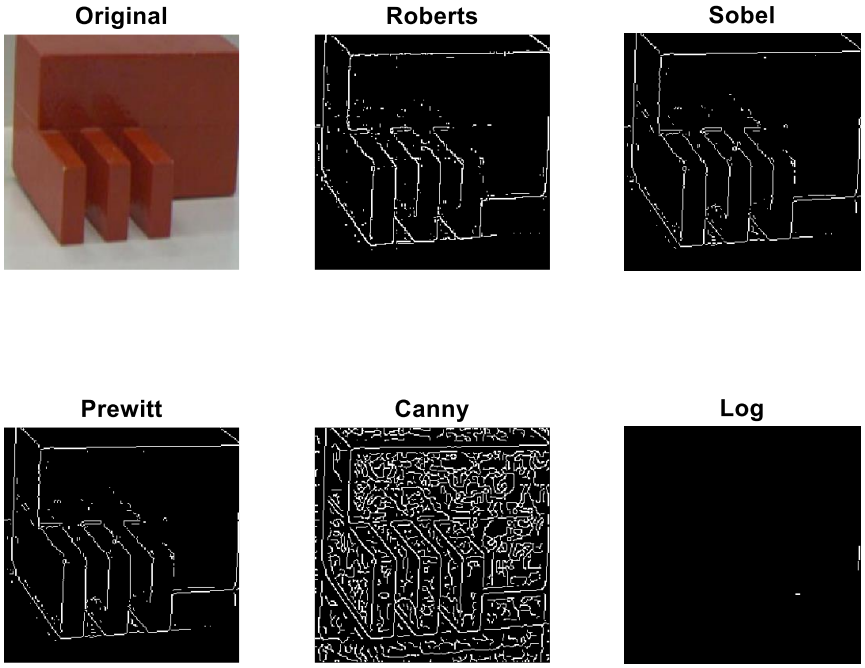
1. Робертсова метода (енгл. *Roberts method*),
2. Собелова метода (енгл. *Sobel method*),
3. Превит метода (енгл. *Prewitt method*),
4. Кани метода (енгл. *Canny method*) и
5. Лапласијан Гаусова метода (енгл. *Laplacian of Gaussian method*).

Процес проналажења ивица контура, математички посматрано, представља процес конволуције. Идеја је да се проласком једног или више филтера кроз слику детектује што више пиксела који припадају ивицама објеката. Класичан приступ је да се на основу информација стечених применом филтера за детекцију хоризонталних, вертикалних и косих ивица формира слика о целокупној контури објеката.

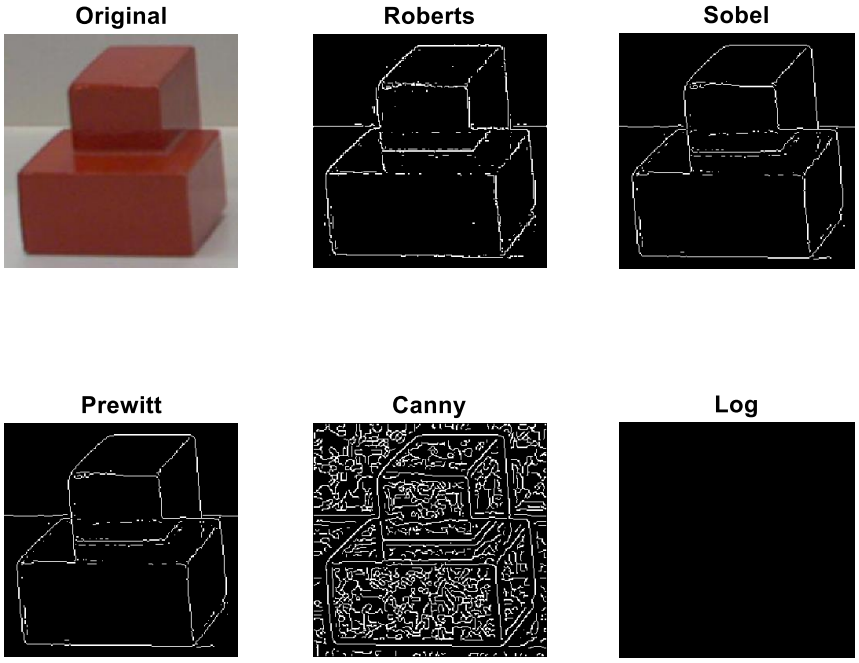
Како свака од напоменутих метода примењује другачије филтере, односно на другачији начин апроксимира функцију осветљености, изглед детектоване контуре објекта је другачија за сваку методу. Не може се експлицитно тврдити која од поменутих метода је најбоља, из разлога што ниједна метода не може да детектује целокупну контуру објеката са слике. Свака метода се у зависности од врсте проблема може успешно применити за детекцију ивица. Како би пронашли методу која највише одговара постављеном проблему, извршено је тестирање поменутих метода са сликама машина алатки које се налазе у лабораторијском моделу технолошког окружења. Приликом тестирања, све методе су тестиране са истим прагом детекције ивица од 0.017. Резултати тестирања, приказани су на сликама 3.5 – 3.8.



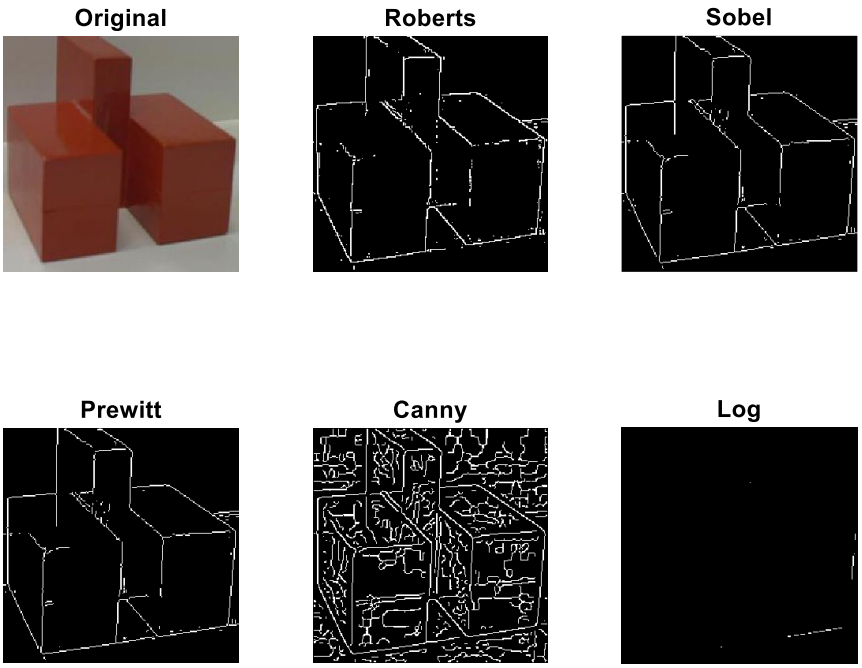
Слика 3.5 – Проналажење ивица машине 1



Слика 3.6 – Проналажење ивица машине 2



Слика 3.7 – Проналажење ивица машине 3



Слика 3.8 – Проналажење ивица машине 4

Треба напоменути да се након сегментације слика применио процес „стањивања“ детектованих линија. На тај начин се прецизније одређују ивични пиксели, чиме се, у зависности од проблема, може знатно олакшати даљи кораци идентификације објеката. На слици 3.9, је дат пример са и без „стањивања“ детектованих линија добијених применом Собелове методе на слику машине 2.

Слика 3.9 – Приказ са и без „стањивања“ детектованих линија

На основу приложеног примећује се да су Робертсова, Собелова, Превитова и Кани метода показале добре резултате приликом детекције ивица објеката, док са друге стране *LoG* метода није успела да детектује ниједну потпуну ивицу. Међутим, иако се Кани методом детектују све ивице објекта, због превеликог шума који настаје, практично је немогуће користити у даљем разматрању.

Међу преостале три методе, након интензивног тренирања и варијације прага, закључено је да сегментација коришћењем Робертсове методе проузрокује недопустиво велики шум, услед детекције бројних кратких ивица које не побољшавају сегментацију целокупне слике. За разлику од ње, Первит метода, у појединим ситуацијама, не успева да детектује круцијалне ивице за процес сегментације, као што се може приметити на слици 3.6 – машина 3.

На основу приложеног, у даљој реализацији овог рада користиће се Собелова метода као „златна средина“ између ове две методе. Детекција ивица Собеловом методом се врши помоћу филтера G_x за детекцију хоризонталних ивица и филтера G_y за детекцију вертикалних ивица.

$$G_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

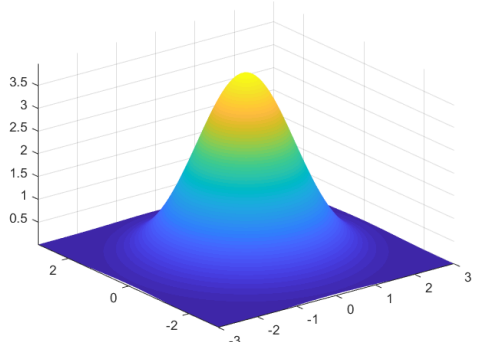
3.4. Примена и значај Гаусовог филтера приликом сегментације слике

Гаусов филтер (слика 3.10) представља вишедимензионалну матрицу чије вредности подлежу нормалној расподели. Примењује се код готово свих алгоритама машинског гледања, у почетним фазама процесирања слике, како би се извршило „замућење“ слике и тиме уједно елиминасао шум и смањио ниво детаља слике.

Вредности матрице Гаусовог филтера, приликом „замућења“ дводимензионалних слика, се добијају на основу једначина 15:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (15)$$

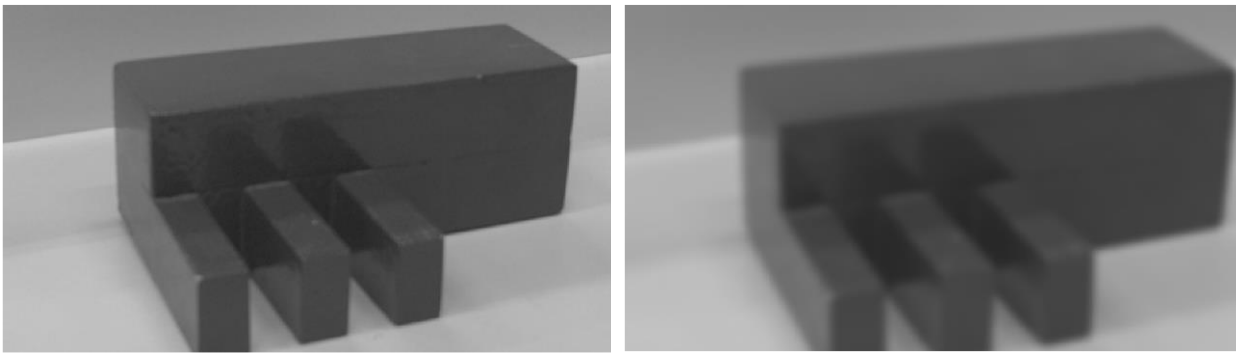
где се са x и y описује позиција пиксела у матрици, а са σ стандардна девијација. Величином филтера се дефинише утицај околних пиксела на „замућење“ пиксела који се тренутно посматра.



Слика 3.10 – Гаусова расподела

Вредности у новодобијеној матрици се, као код конволуције, добијају сумирањем производа пиксела филтера са коресподентним пикселима слике. Међутим, за разлику од процеса конволуције, централни пиксел Гаусовог филтера мора да прође кроз сваки пиксел слике како би замућење било уједначено. Да би то било могуће, потребно је решити проблем преласка пиксела филтера приликом замућења ивичних пиксела слике. Тај проблем се ефикасно решава додавањем пиксела око ивица слике (енгл. *Image Padding)*, при чему је број слојева додатих пиксела једнак максималном броју пискела који прелазе слику дуж једне осе. Додати пиксели узимају вредности последњег пиксела слике дуж исте осе.

На слици 3.11, представљен је случај пре и након примене Гаусовог филтера. У овом примеру, како би замућење слике дошло до изражаја, коришћен је филтер са већом стандардном девијацијом него приликом експлоатације интелигентног мобилног робота RAICO. Након примене Гаусовог филтера, да се приметити да је ниво рефлексије светлости смањен, што знатно олакшава „посао“ у даљем току процесирања слике.



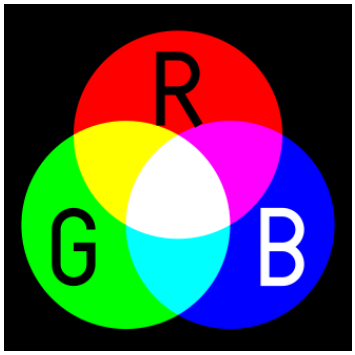
Слика 3.11 – Слика пре (лево) и након (десно) примене Гаусовог фитлера

3.5 Анализа различитих модела репрезентације слике у боји и њихова примена у сегментацији слике

Након аквизиције слике са камере, добијена дигитална слика се може представити различитим моделима, међу којима у домену машинског гледања доминирају RGB и HSV модел. Репрезентација боје пиксела слике се састоји из три вредности, које код RGB модела представљају следствено вредности интензитета осветљености црвене, зелене и плаве боје, док код HSV модела вредности за нијансу (*Hue)*, сатурацију (*Saturation)* и интензитет (*Value)*.

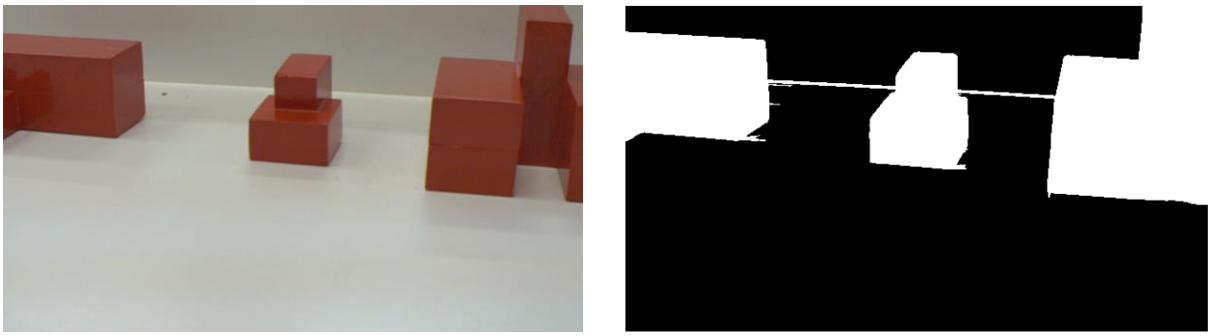
RGB модел репрезентације боје на слици

RGB модел репрезентације боје (слика 3.12) представља један адитивни модел репрезентације боја, код кога се мешањем нијанси црвене, зелене и плаве боје, по којима је модел добио име, формира читав спектар боја. Због своје једноставности, RGB модел је у данашњици пронашао велику примену, и то првенствено у очитавању, представљању и приказивању слика у електронским системима, попут телевизора, рачунара, мобилних телефона итд. Репрезентација боје не зависи само од интензитета осветљености RGB боје, већ и од самог уређаја за репрезентацију. Наиме, сваки електронски уређај поседује различите јединице за представљање боје, па ће самим тим и репрезентација дате вредности RGB боје бити другачија.



Слика 3.12 – RGB модел репрезентације слике у боји [14]

У компјутерској графици, вредности интензитета осветљености боја се крећу од 0 до 255. Уколико су све вредности интензитета осветљености 0, приказаће се црна боја, док у супротном, уколико су њихове вредности 255, приказаће се бела боја. Променом нумеричких вредности интензитета осветљености поменутих боја формирају се различите нијансе свих осталих боја.

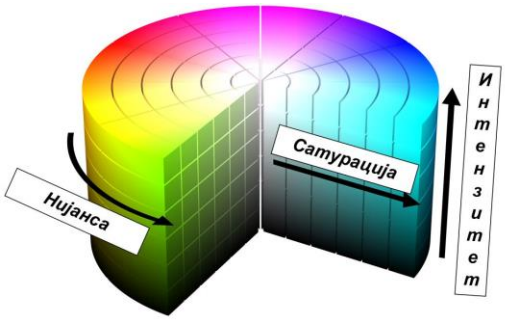


Слика 3.13 – Слике пре и након примене RGB прага

Знајући RGB вредности свих пиксела са слике, користећи одговарајући праг (енгл. *Treshold*), може се извршити сепарација једнобојних објеката (слика 3.13). Пошто су сви објекети у постављеном проблему црвене боје, а околина беле боје, применом само једног прага извршава се детекција ликова објеката на слици. Због ниске вредности интензитета осветљености зелене боје у репрезентованој црвеној боји објеката, приликом детекције ликова објеката примењује се праг за вредност интензитета осветљености од 120.

HSV модел репрезентације боје на слици

HSV модел репрезентације боје представља један интуитивни модел, код кога се формирање боја врши на основу вредности за нијансу (енгл. *Hue)*, сатурацију (енгл. *Saturation)* и интензитет (енгл. *Value)*. Модел је пројектован крајем 20. века, од стране инжењера компјутерске графике, са циљем да се репрезентација боје на електронским уређајима приближи људској перцепцији. HSV модел се визуелно може представити у облику цилиндара (слика 3.14), при чему се репрезентација боје врши на основу следећих компоненти:



Слика 3.14 – HSV модел репрезентације слике у боји [15]

1. ***Нијанса (енгл. Hue)*** представља радијалну променљиву, око централне осе цилиндра, којом се описује палета боја. Пошто је у питању радијална променљива, *Hue* може узети вредности између 0° и 360°, при чему се црвена боја описује вредностима између 0° и 60°, зелена боја вредностима између од 120° и 180° и плава боја вредностима између 240° и 300°. Променом угла врши се мешање ових боја у одређеној количини, и добијање жељене нијансе боје.

2. ***Сатурација (енгл. Saturation)*** представља променљиву којом се описује ниво израженості боја. Може узети вредности између 0 – 100%, при чему се са вредоншћу сатурације од 0% описује бледа боја, попут сиве, а са вредношћу од 100% „живахна“ боја одређене нијансе.

3. ***Интензитет (енгл. Value)*** представља компоненту којом се описује интензитет осветљености боја. Као и сатурација, може имати вредности између 0 – 100%, где се са 0% описује потпуно црна боја, а са 100% потпуно осветљена нијансе одређене боје.

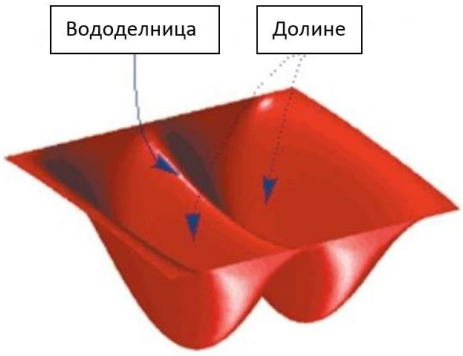
У претходном случају, применом RGB прага успешно су раздвојени региони који припадају објектима од региона окружења. Међутим, како би се формирала коначна слика ликова објеката, потребно је на адекватан начин елиминисати ивице између зида и пода (слика 3.15). Експерименталним испитивањима се утврдило да се применом HSV прага сатурације (вредности 0.26) остварују најбољи резултати, при чему су се анализирали само детектовани пиксели претходно добијене бинарне слике.

Слика 3.15 – Слика пре и након примене HSV прага

3.6 Сепарација објеката применом „Вододелница“ алгоритма

Инспирација за пројектовање алгоритма дошла је од појма вододелница, по коме је алгоритам и добио име. Вододелница представља замишљену границу од које се све воде, које падају на земљу, сливају у два различита речна система. Због своје једноставности и високе поузданости, алгоритам проналази велику примену код многих визуелних система препознавања, поготово у области медицине.

Приликом коришћења алгоритма, слика је посматрана као топографски пејзаж (слика 3.16) са доста „долина“ (енгл. *Catchment Basians*), при чему се „дубина“ пиксела описује њиховим интензитетом осветљености сиве боје или интензитетом градијента функције. Идеја алгоритма јесте да се пронађу „вододелнице“ (енгл. *Watershed Lines*) које би на основу „најдубљих“ тачака „долина“, односно локалних минимума, на адекватан начин извршиле раздвајање преклапајућих објеката на слици. Сепарација објеката применом „Вододелница“ алгоритма је извршена коришћењем модификоване верзије алгоритма објављеног на *web* страници [16].

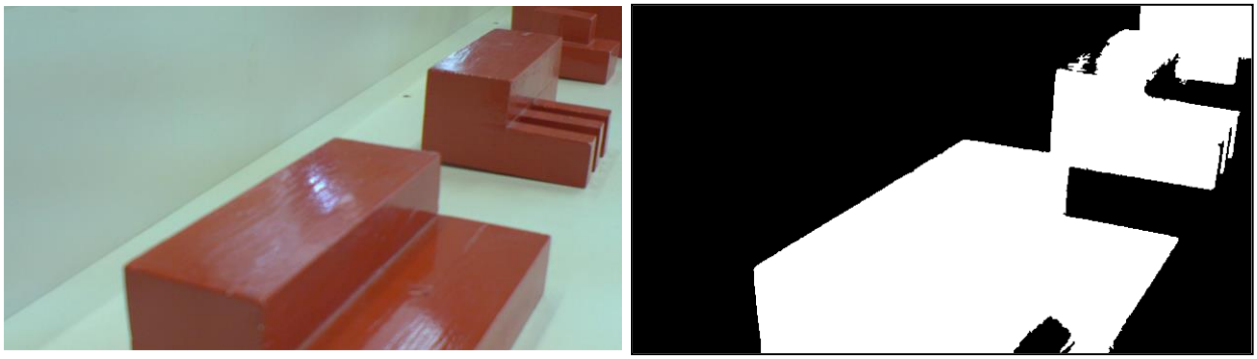


Слика 3.16 – Приказ сепарације „долина“ помоћи вододелница [30]

Функционалност „Вододелница“ алгоритма се може описати следећим корацима:

1. *Учитавање и процесирање слика*

Да би алгоритам могао да се примени, потребно је извршити аквизицију слика са камера и применити претходно поменуте алгоритме, како би се добила бинарна слика са детектованим ликовима објеката (слика 3.17).



Слика 3.17 – Слика добијена са камере (лево); Бинарна слика добијена процесирањем (десно)

2. *Одређивање локалних минимума*

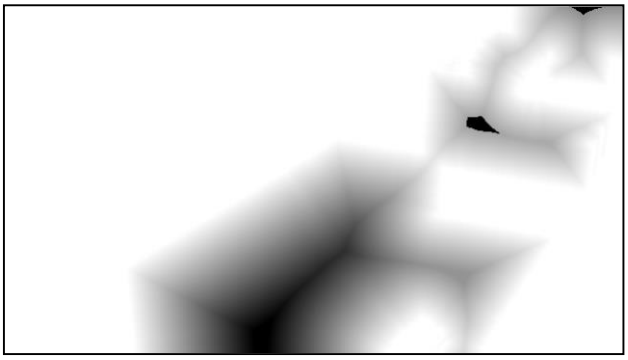
Локални минимуми (слика 3.18) се дефинишу као групе пиксела, са интензитетом осветљености 1, који имају највећу Еуклидску удаљеност од пикселса са интензитетом осветљености 0. На основу њих алгоритам има представу о томе колико се „долина“, односно објеката, налази на слици. Из тог разлога, одређивање локалних минимума представља кључни корак приликом сепарације преклапајућих објеката.

Уколико постоји више локалних минимума који припадају истом објекту, може доћи до стварања додатних „долина“, што би проузроковало и стварање додатних „вододелница“, које би извршиле непожељну сепарацију једног објеката.

Слика 3.18 – Приказ локалних минимума бинарне слике

3. Формирање мапе удаљености пиксела

Мапа удаљености пиксела (слика 3.19) представља матрицу у којој су, на местима пиксела са интензитетом осветљености 1, уписане вредности Еуклидске удаљености посматраног пиксела од најближег пиксела са вредношћу интензитета осветљености 0. Како би се пиксели локалног минимума, као најудаљенији пиксели, разликовали од осталих пиксела, за њихове вредности у мапи удаљености додељује им се вредност 255. На основу мапе удаљености алгоритам прима информацију о „нагибу долине“, која се користи приликом одређивања положаја „вододелница“.



Слика 3.19 – Мапа удаљености пиксела

4. Проналажење „вододелница“

Поступак проналажења „вододелница“ је инспирисан следећим биолошким процесом: над поменутим топографским пејзажом (мапа удаљености) пада киша, при чему долази до „попуњавања долина водом“ и евентуалног „преливања“. Вододелница се поставља на оним положајима где вода почиње да прелива из једне „долине“ у другу.

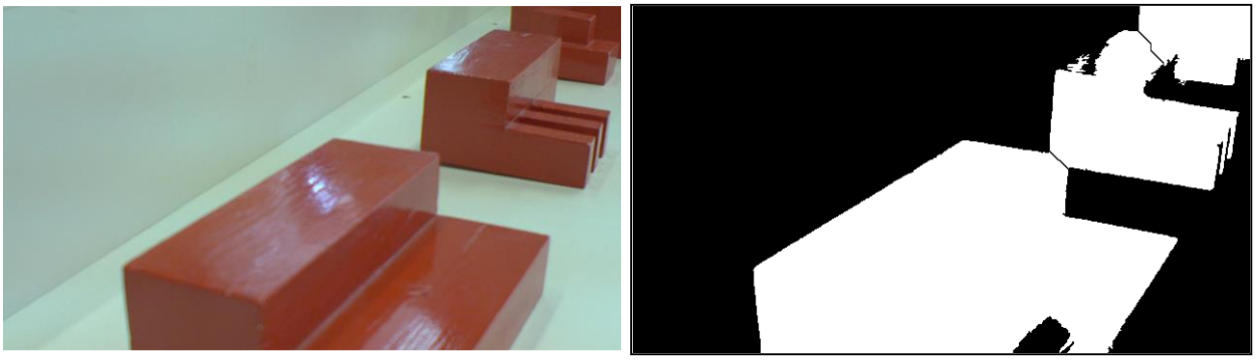
Дакле, положај „вододелница“ претежно зависи од позиције локалних минимума и од „нагиба долина“. У областима где нема „долина“, „вододелница“ наставља да прати формирану границу преливања до краја оквира слике.

На слици 3.20, је представљен резултат примене „Вододелница“ алгоритма, при чему су „вододелнице“ означене белом бојом, а подељени региони различитим бојама. Пошто „Вододелнице“ представљају граничне прелазе између региона, њихова дебљина износи 1 пиксел, док су на слици 3.20 представљене наглашеном дебљином ради боље визуализације. Свака боја представља одређену „долину“, при чему, као што је већ наглашено, број долина зависи од броја детектованих локалних минимума.

Слика 3.20 – Резултат примене „Вододелница“ алгоритма

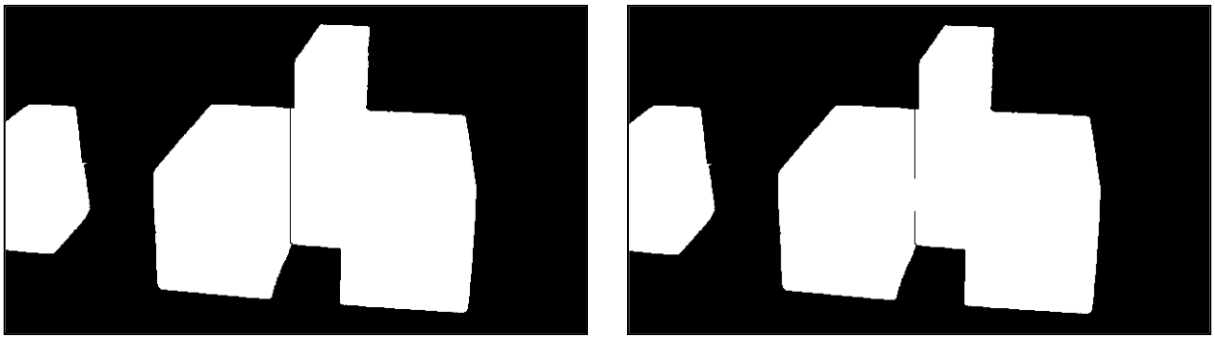
5. Сепарација преклапајућих објеката

Након проналаска „вододелница“ финални корак алгоритма јесте сепарација објеката. Сепарација се извршава уклањањем пиксела који уједно припадају објектима и „вододелници“. Коначан резултат сепарације објеката применом „Вододелница“ алгоритма је приказан на слици 3.21.



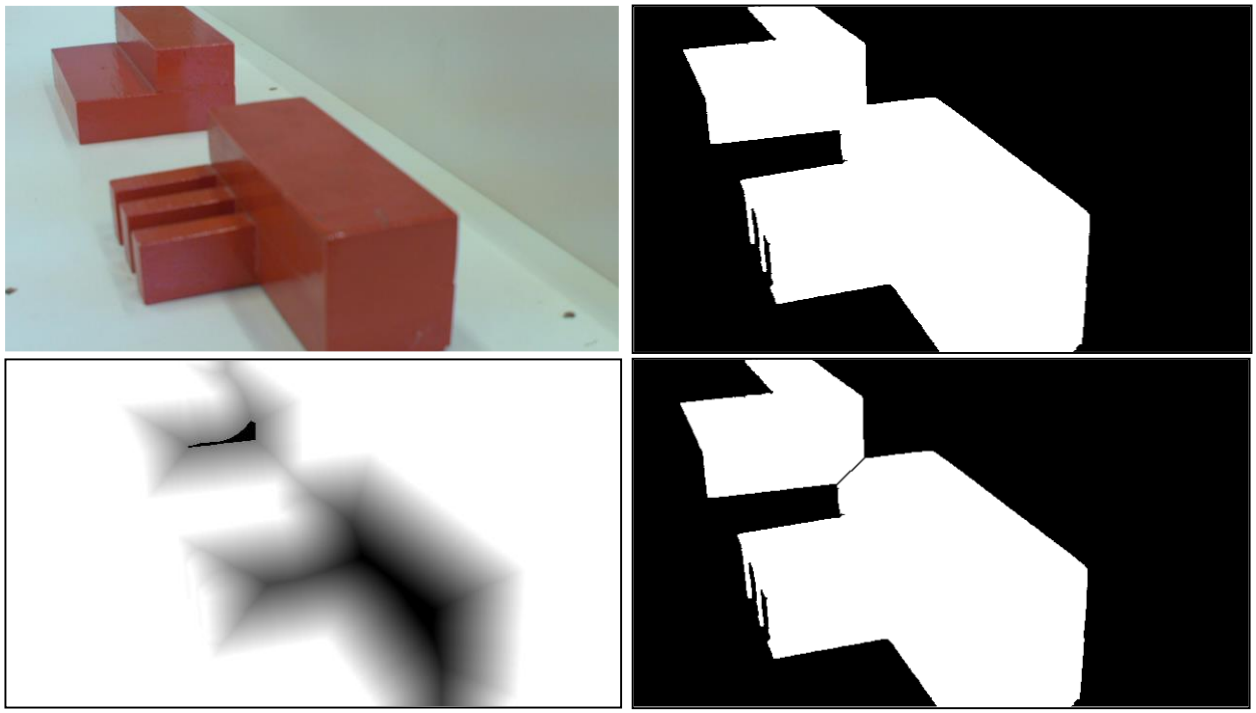
Слика 3.21 – Сепарација објеката применом „Вододелница“ алгоритма

У склопу главног алгоритма, пројектован је алгоритам на основу кога се извршава провера и корекција грешке „Вододелница“ алгоритма. Грешке су примарно проузроковане усед појаве сувишних локалних минимума. Применом CNN-а, извршава се идентификација тренутних ликова објеката на слици, при чему уколико се примети да постоје више објеката који припадају истој класи, и који се налазе у непосредној близини, извршава се њихово спајање. На слици 3.22, приказана је машина 4 коју је „Вододелница“ алгоритам неисправно раздвојио. Након идентификације раздвојених делова утврђено је да региони припадају истом објекту и извршено је њихово спајање. Резултат спајања приказан је на слици 3.22.

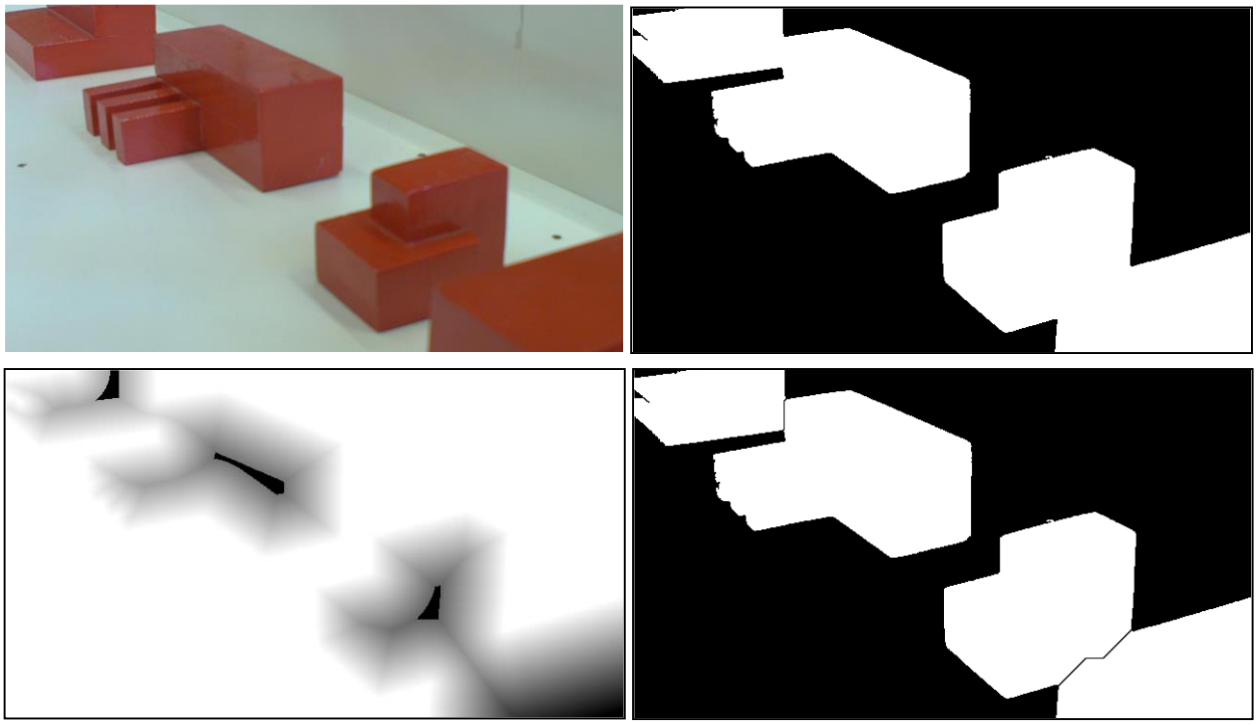


Слика 3.22 – Приказ пре и након извршавања корекције „Вододелница“ алгоритма

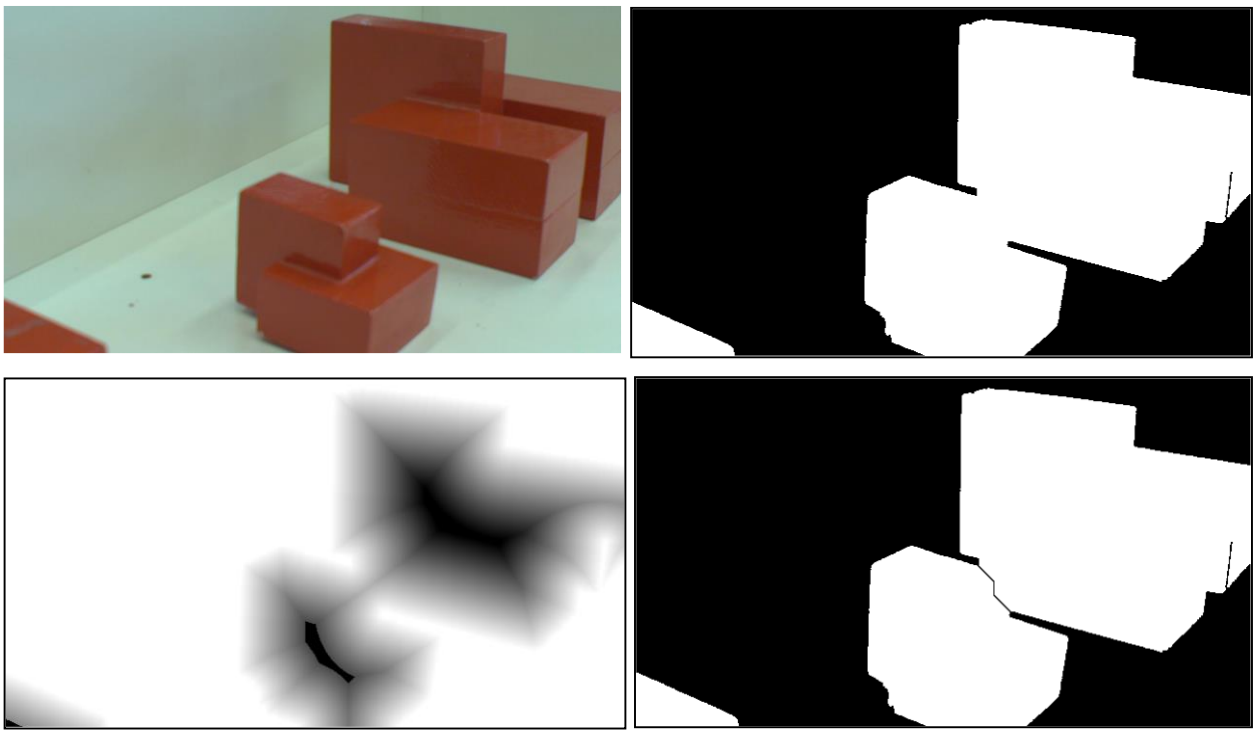
На сликама 3.23 – 3.26, приказани су резултати примене „Вододелница“ алгоритма приликом експлоатације интелигентног мобилног робота RAICO у лабораторијском моделу технолошког окружења.



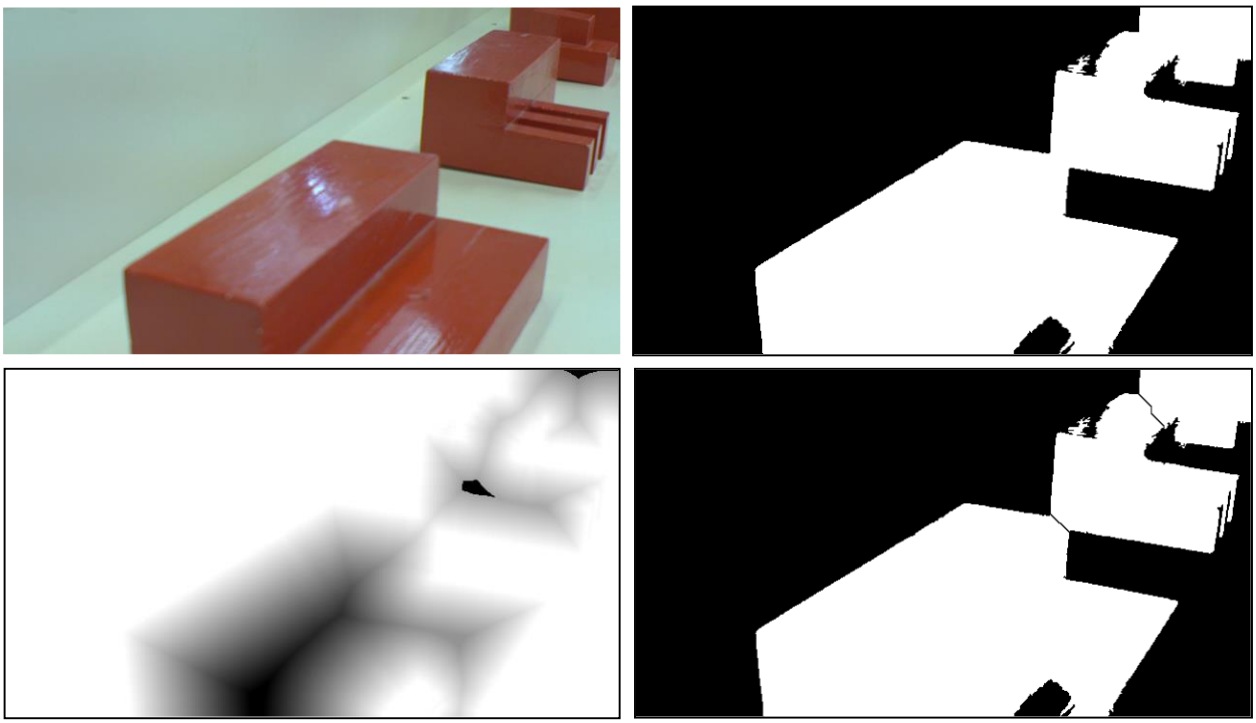
Слика 3.23 – Пример „Вододелница“ алгоритма бр.1



Слика 3.24 – Пример „Вододелница“ алгоритма бр.2



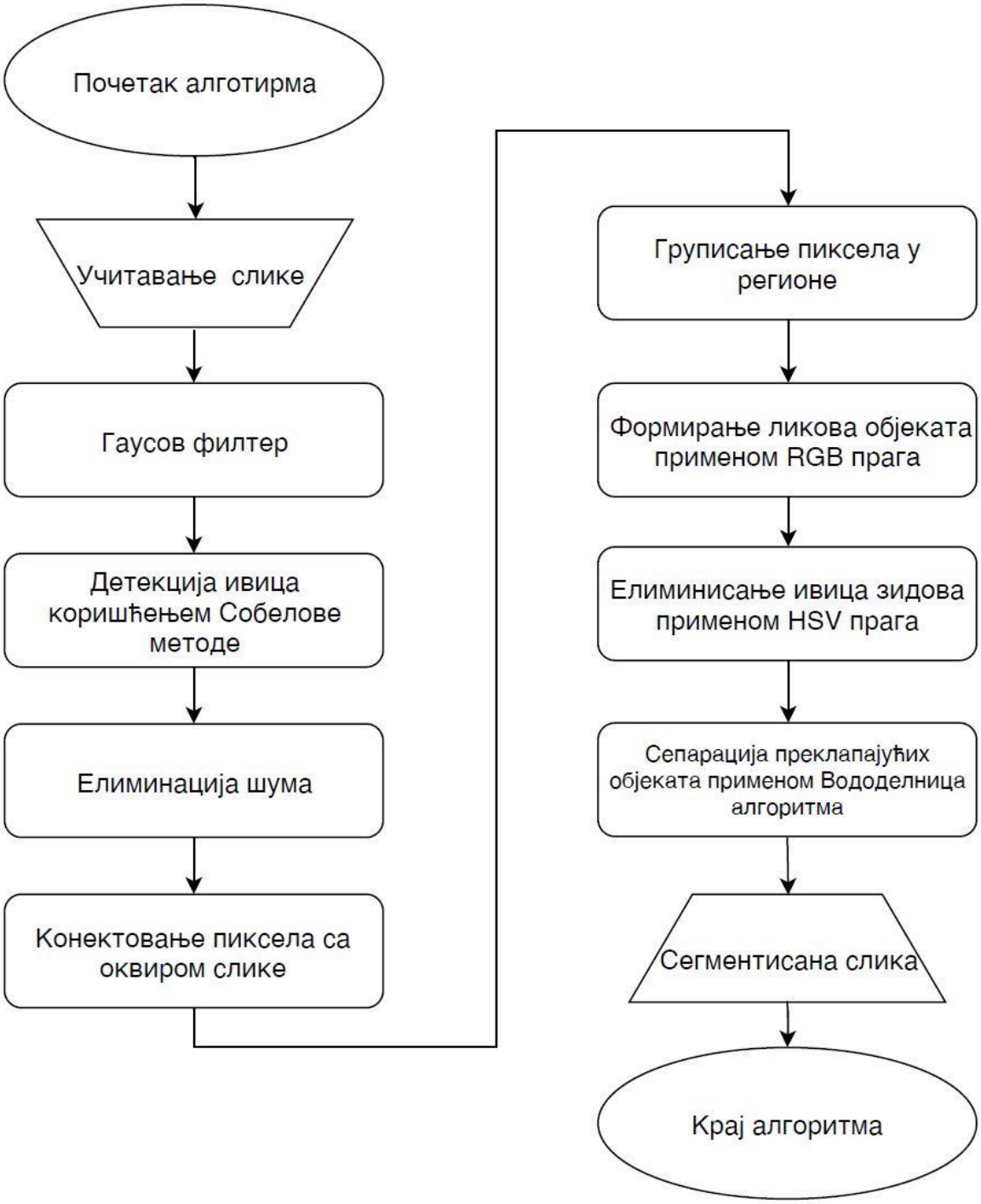
Слика 3.25 – Пример „Вододелница“ алгоритма бр.3



Слика 3.26 – Пример „Вододелница“ алгоритма бр.4

4. Модул за детекцију објеката применом метода сегментације слике базиране на детекцији ивица

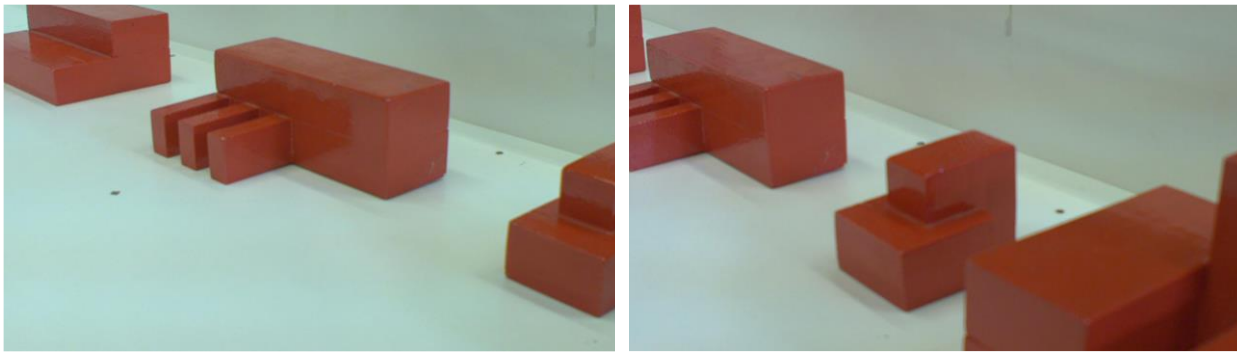
У оквиру овог поглавља, биће приказани и детаљно описани кораци приликом детекције ликова објеката на слици. Алгоритам по коме се извршава детекција ликова објеката је приказан на слици 4.1.



Слика 4.1 – Алгоритам за детекцију ликова објеката на слици

1. Учитавање слике

На слици 4.2, приказане су слике добијене током експлоатације интелигентног мобилног робота RAICO, унутар лабораторијског модела технолошког окружења, коришћењем две Basler acA1920-25uc камере. Пре аквизиције података, извршено је подешавање жељеног нивоа осветљености (енгл. „*White Balance*“) и фокуса.

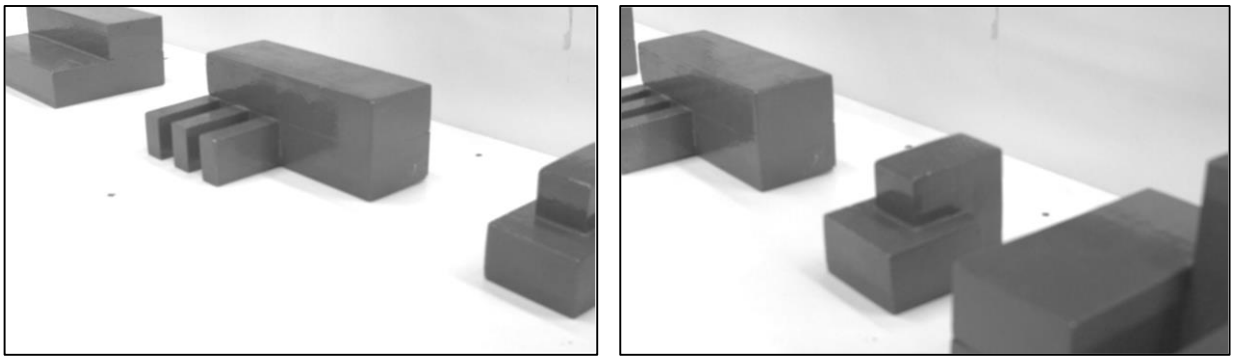


Слика 4.2 – Слике добијене аквизицијом података са леве и десне камере

2. Примена Гаусовог филтера

Након аквизиције слика са камера, следећи корак приликом детекције ликова објеката јесте примена Гаусовог филтера. Добијене слике са камера (слика 4.2) су, ради убрзавања процеса конволуције, трансформисане у монохроматски облик након чега је над њима примењен Гаусов филтер, величине 3x3 пиксела. Резултат примене Гаусовог филтера је представљен на слици 4.3, где се уочава да је поред малог „замућења“ слике дошло до повећања интензитета осветљености услед значајних вредности примењеног филтера. Тиме је контраст између објеката и окружења дошао до већег изражаја, чиме се знатно олакшало проналажење ивица објеката.

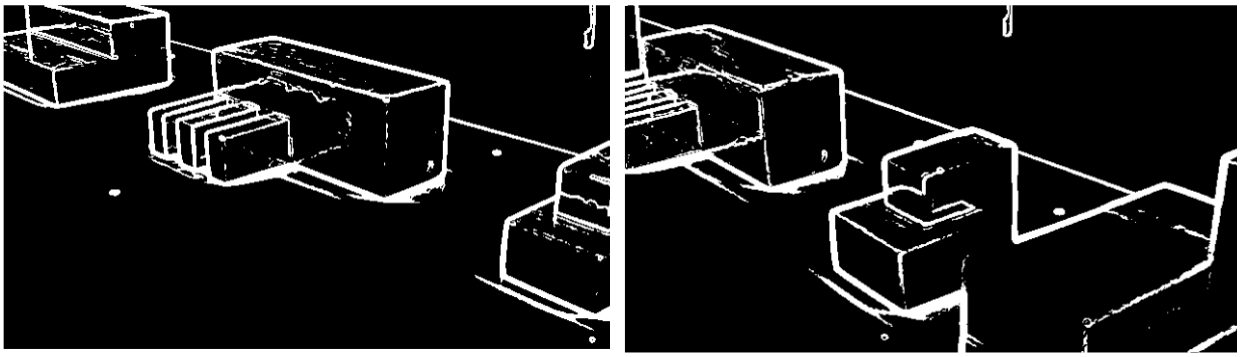
Због немогућности филтера да прође кроз извичне пикселе добијених слика, дошло је до смањења димензија слике, за по један пиксел по свакој оси. Тај проблем се решио дуплирањем последњег реда, односно колоне пиксела новодобијене слике.



Слика 4.3 – Слике добијене проласког Гаусовог филтера

3. Детекција ивица коришћењем Собелове методе

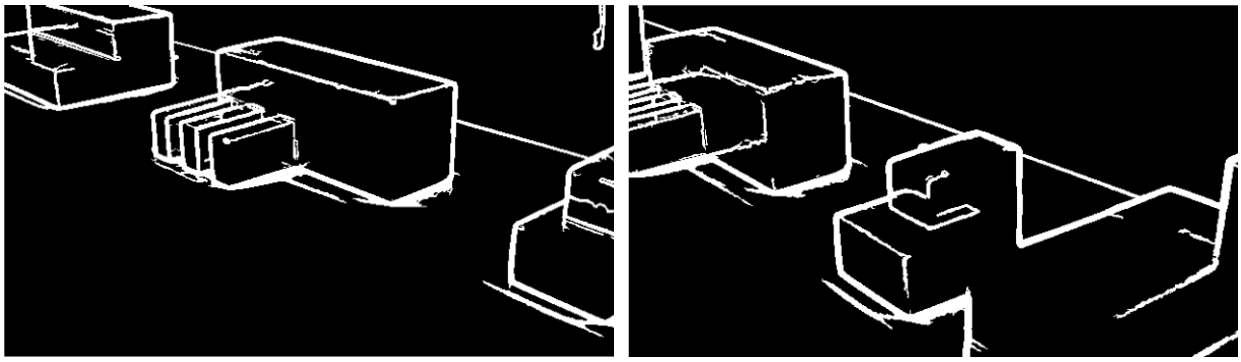
Следећи корак јесте да се на основу слика, добијених применом Гаусовог филтера, изврши детекција ивица коришћењем Собелове методе, док праг за детекцију ивица износи 0.017. Како би се добиле што израженије ивице објеката, а тиме и унапредило формирање затворених контура, након детекције ивица није извршено њихово „стањивање“. Резултати детекције ивица, применом Собелове методе, приказани су на слици 4.4.



Слика 4.4 – Детекција ивица коришћењем Собелове методе

4. Елиминација шума

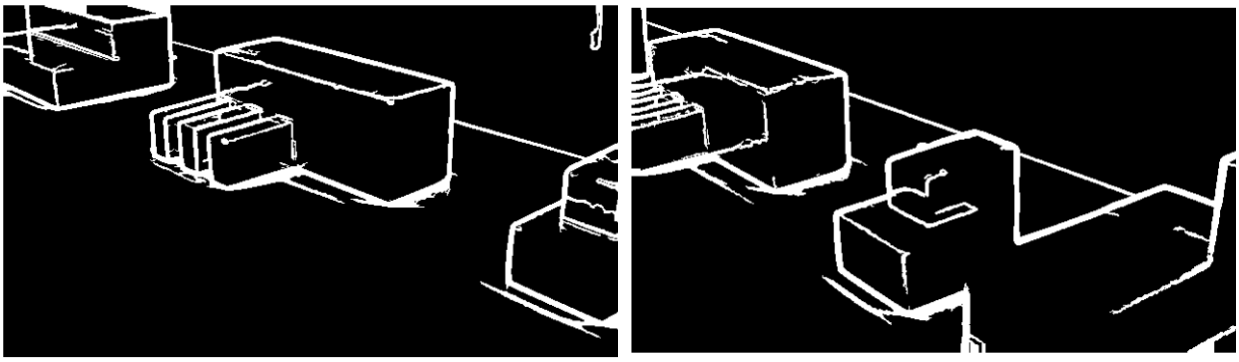
Као што се може приметити на слици 4.4, приликом детекције ивица долази до настанка шума који може изазвати проблеме у наредним корацима. Под шумом се у овом случају подразумевају сви пиксели, односно групације пиксела, који нису конектовани са ивичним пикселима објеката или зида. Елиминација се извршава уколико групација пиксела има мање чланова од дефинисаног прага, који износи 300 пиксела. Резултат елиминације шума приказан је на слици 4.5.



Слика 4.5 – Елиминисање шума

5.Конектовање пиксела са оквиром слике

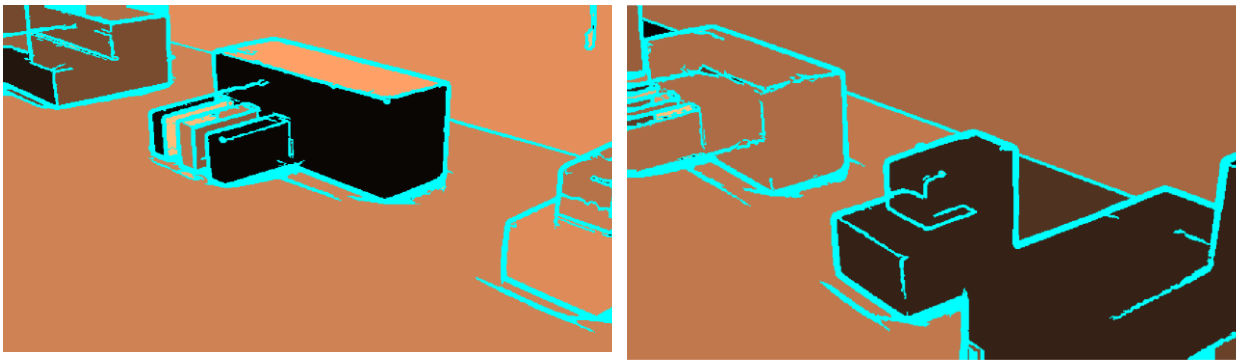
Овим кораком, извршава се спајање ивица детектованих објеката и оквира слике, уколико се те ивице налазе у близини оквира слике. На тај начин се обезбеђује затвореност контура на слици, чиме се унапређује даљи процес сегментације. Спајање се извршава претраживањем првих и последњих десет врста и колона слике, при чему се у случају детекције пиксела са вредношћу интензитета осветљености 1, извршава попуњавање свих пиксела између детектованог пиксела и најближе ивице оквира слике. Пошто су сви ивични пиксели објеката на слици 4.5 у близини оквира слике успешно детектовани, проласком слике кроз овај корак неће имати значајан утицај на даљи ток сегментације слике.



Слика 4.6 – Конектовање пиксела са оквиром слике

6. Груписање пиксела у регионе (енгл. Region Clustering)

Груписање пиксела се извршава на основу детектованих ивица, тако што се пиксели унутар истих затворених контура посматрају као једна целина, односно регион. Након груписања, сваки пиксел на слици припада одређеном региону или ивици региона. Када су познати региони, могуће је издвојити њихове карактеристике као што су нпр.: списак пиксела (енгл. *PixelList*), површина региона (енгл. *Area*), тежиште региона (енгл. *Centroid*), минимална, средња и максимална вредност интензитета нивоа осветљености региона итд. На слици 4.7, приказан је резултат груписања пиксела, при чему је сваки детектовани регион обојен другом бојом.

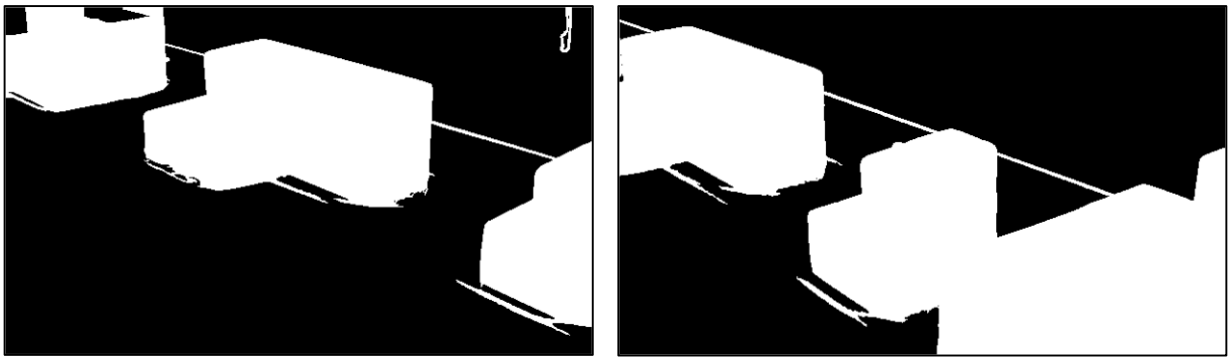


Слика 4.7 – Груписање пиксела у регионе

7. Формирање ликова објеката применом RGB прага

Следећи корак јесте да се на основу издвојених карактеристика региона, из претходног корака, изврши селекција региона који припадају објектима. Селекција се може извршити на више различитих начина, при чему се у овом раду, због велике разлике између црвене боје објеката и беле боје окружења, примењује RGB праг.

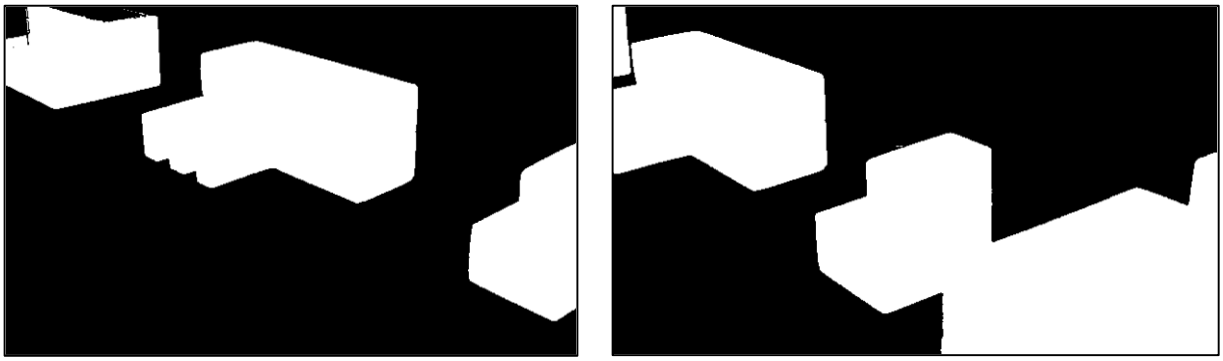
Како се у црвеној боји налази мала количина интензитета осветљености зелене боје, приликом формирања ликова објеката, за праг се узима средња вредност зелене боје целог региона од 120. Дакле, региони који имају мању средњу вредност интензитета осветљености зелене боје од постављеног прага се сматрају ликовима објектима, док се остали региони сматрају окружењем. Међутим, овим приступом се посматрају само региони, а не и ивице региона, што значи да су детектоване ивице, након извшене RGB анализе, и даље видљиве (слика 4.8).



Слика 4.8 – Формирање ликова објеката применом RGB прага

8. Елиминисање ивица зидова окружења применом HSV прага

Један од начина за уклањање заосталих непожељних детектованих ивица јесте применом HSV анализе. Како би се убрзало време процесирања, анализирају се само детектовани пиксели бинарне слике. Варирањем вредности прагова за нијансу боје (енгл. *Hue*), сатурацију (енгл. *Saturation*) и интензитет (енгл. *Value*), дошло се до закључка да је за елиминацију ивица зидова оптимално решење користити праг сатурације од 0.26. Елиминација се извршава променом интензитета осветљености пиксела бинарне слике из 1 у 0, уколико је вредност сатурације мања од задате вредности прага. На овај начин се поред елиминације ивица зидова окружења на ефективан начин елиминише и заостали шум. Резултат примене алгоритма на претходно добијену слику 4.8, приказан је на слици 4.9.



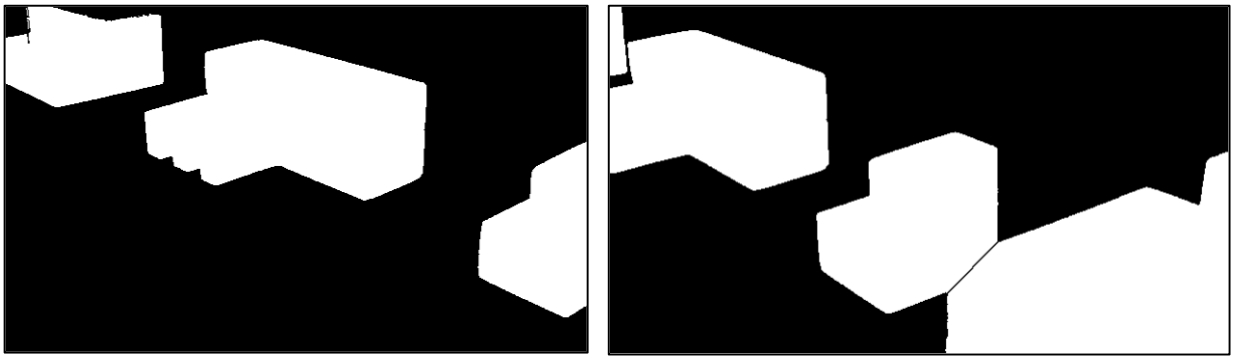
Слика 4.9 – Елиминисање ивица зидова применом HSV прага

9. Сепарација преклапајућих објеката применом „Вододелница“ *алгоритма*

Приликом кретања интелигентног мобилног робота RAICO кроз лабораторијски модел технолошког окружења, постоји могућност да се на слици детектују објекти који се

преклапају. Сегментацијом такве слике, применом претходних корака, би се успешно раздвојили ликови објеката од позадине, али не и један од другог (слика 4.9).

Како би се решио тај проблем, потребно је увести алгоритам за сепарацију преклапајућих објеката, као што је нпр. „Вододелница“ алгоритам. Принцип функционисања алгоритма је објашњен у поглављу 3.6, а резултати његове примене су приказане на слици 4.10.

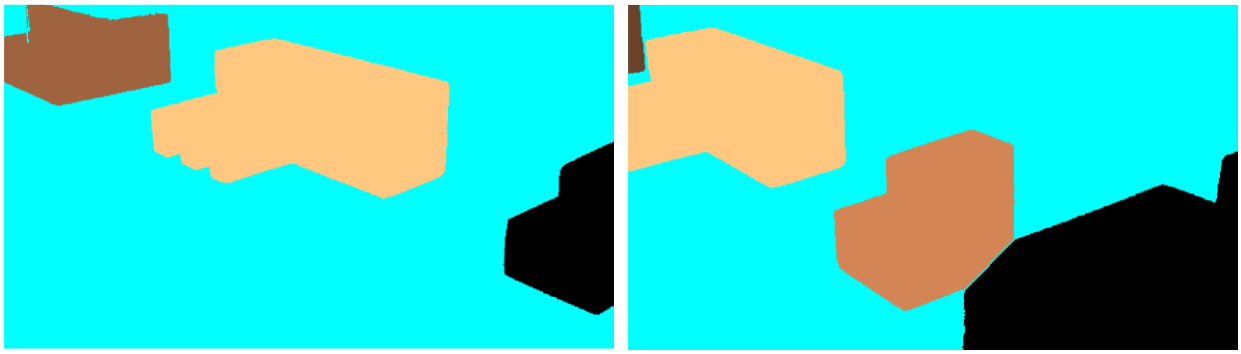


Слика 4.10 – Сепарација преклапајућих објеката применом „Вододелница“ алгоритма

10. Формирање и репрезентације сегментисане слике

У последњем кораку, преостало је да се на основу добијене бинарне слике 4.10 формира коначан облик сегментисане слике. Како су региони слике који припадају ликовима објеката добијени претходним корацима, формирање сегментисане слике се извршава индексирањем детектованих пиксела, тако да сви пиксели унутар истог региона поседују исту идентификациону ознаку.

Резултат примене алгоритма за детекцију технолошких ентитета над сликама 4.2, приказан је на слици 4.11.



Слика 4.11 – Коначан облик сегментисане слике

5. Идентификација детектованих ликова објеката

5.1 „Дообучавање“ вештачких неуронских мрежа (енгл. Transfer Learning)

Ради смањења времена обучавања ВНМ, као алтернатива класичном приступу обучавања може се применити процес „дообучавања“. „Дообучавање“ (слика 5.1) представља поступак поновног тренирања ВНМ са новим обучавајућим паровим при чему је пре тренирања потребно извршити модификацију излазних слојева да одговара постављеном проблему. Приликом израде овог рада разматрали су се обучени модели CNN-а, који су остварили значајне резултате у домену идентификације објеката у претходним годинама.

Доступне мреже које су се разматране су:

- *AlexNet [17],*
- *GoogleNet [18],*
- *ResNet – 18 [19]* и
- *ResNet – 50 [19].*

Слика 5.1 – Шема „дообучавања“ мреже (енгл. Transfer Learning) [20]

Поменуте мреже су истрениране са преко милион слика из *ImageNet* [21] базе података, да класификују 1000 различитих објеката. Како у нашем случају постоје 4 класе објеката које треба идентификовати, потребно је извршити модификацију последњег потпуно повезаног слоја (енгл. *Fully Connected layer*) и излазног слоја (енг. *Output layer*) мреже, тако да њен излаз представља вектор вероватноће тих класа. Након тога мрежа је спремна за „дообучавање“ тј. поновно тренирање, са прикупљеним обучавајућим паровима, односно сликама објеката које треба идентификовати.

Обучавајући парови су се прикупили тако што се мобилни робот постављао у карактеристичне положаје након чега му се задавала команда за кретање. При сваком кретању прикупљено је 50 слика са леве и 50 слика са десне камере. Те слике, у наставку, коришћене су како би се процесом сегментације генерисало укупно 400 засебних слика објеката које, као што је већ напоменуто, представљају обучавајуће парове. Све мреже су трениране са истим скупом обучавајућих парова, при чему се пре тренирања насумично бира 70% за тренирање и 20% за тестирање и 10% за валидацију. Тренирање мрежа је извршено коришћењем алгоритма објављеног на *web* страници [33].

Предност „дообучавања“, у односу на обучавање нових мрежа, јесте у томе што су тежински односи већ подешени да препознају одређене карактеристике објеката као што су хоризонталне, вертикално и косе ивице, а самим тим и карактеристичне облике као што су круг, квадрат и троугао итд. Даљом оптимизацијом тежинских односа мрежа почиње да препознаје детаљније карактеристике објеката, што условљава већу тачност при њиховој идентификацији.

Пре него што се ступи у тренирање мреже потребно је пронаћи оптималне параметре тренирања за решавање постављеног проблема. Како би се то извело, тренирање мреже је извршено из више понављања са варијацијом следећих параметара:

- ***'SolverName'*** - алгоритам обучавања,
- **'*InitialLearnRate*'** - почетни параметар учења. Уколико је параметар учења изразито мали тренирање ће трајати предуго, док са друге стране уколико је превелик мрежа може „прескочити“ глобални минимум функције циља,
- **'*LearnRateDropFactor*' -** коефицијент смањења параметра учења кроз епохе,
- **'*MaxEpochs*'** - максимални број епоха при тренирању,
- **'*MiniBatchSize*'** - величина подскупа слика у једној итерацији. Функција циља се израчунава за све слике у подскупу при чему се на основу сумиране грешке врши промена тежинских односа.

Идеја је да се постигне што краће време обучавања како би се тестирање извршило у што више понављања. Из тог разлога приликом обучавања коришћена је мрежа са најмањим бројем слојева, која у овом случају представља *AlexNet.* Током претходних тестирања, уочило се да мрежа постиже високу тачност већ после друге епохе, па је из тог разлога, како би се додатно смањило време обучавања, број епоха подешен на 2. Тренирање мреже је извршено више пута са истим параметрима, при чему су најбољи остварени резултати приказани у табели 4.

Бр.	Алгоритам обучавања	Почетни пар. учења	Коеф. смањења пар. учења	Величина подскупа слика по итерацији	Време тренирања	Тачност
1.	'sgdm'	0.001	0.2	100	12 min 28 sec	0.8675
2.	'adam'	0.001	0.2	100	13 min 13 sec	0.2506
3.	'adam'	0.001	0.5	100	15 min 5 sec	0.5364
4.	'adam'	0.005	0.5	100	13 min 13 sec	0.2969
5.	'rmsprop'	0.001	0.2	100	12 min 49 sec	0.2693
6.	'sgdm'	0.005	0.2	100	12 min 15 sec	0.7649
7.	'sgdm'	0.008	0.2	100	13 min 17 sec	0.1464
8.	'sgdm'	0.01	0.2	100	11 min 35 sec	0
9.	'sgdm'	0.001	0.3	100	12 min 14 sec	0.9614
10.	'sgdm'	0.001	0.5	100	15 min 46 sec	0.8675
11.	'sgdm'	0.001	0.9	100	15 min 25 sec	0.9481
12.	'sgdm'	0.001	0.2	50	16 min 06 sec	0.9316
13.	'sgdm'	0.001	0.2	200	9 min 55 sec	0.9459
14.	'sgdm'	0.001	0.2	500	11 min 22 sec	0.9294

Табела 4 – Проналажење оптимачних параметара тренирања

Из табеле 4 се може закључити следеће:

- стохастички градијенти поступак са моментом (‘sgdm’) се показао као најбољи алгоритам обучавања,
- тачност драстично опада са вредностима параметра учења изнад 0.005,
- промена коефицијента смањења параметра учења нема значајан утицај на тачност,
- промена величине подскупа слика нема велики утицај на тачност, али зато утиче на време тренирања мреже.

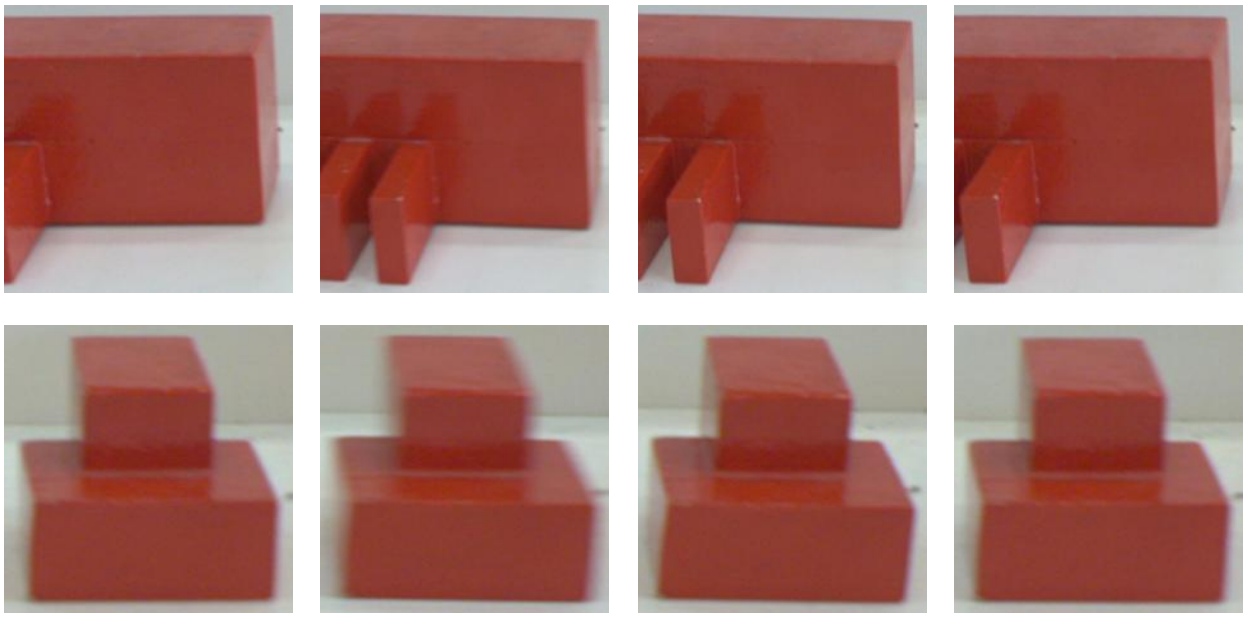
На основу табеле 4, узимајући у обзир остварену тачност и време тренирања, за „дообучавање“ поменутих модела CNN-а користиће се параметри мреже под редним бројем 9, са максимални број епоха подешеним на 5.

Како су слике, из којих су добијени обучавајући парови, прикупљене између малих временитих интервала кретања робота, степен сличности између појединих обучавајућих парова је велики. Из тог разлога тестирање приликом процеса обучавања не приказује реалну слику тачности извршавања алгоритма.

На слици 5.2, приказан је пример „сличних“ обучавајућих парова машине 2 и машине 3, где се примећује да иако је робот извршио промену положаја, и тиме променио своје видно поље, није дошло до великих промена карактеристика (енгл. *Feature*) између прикупљених обучавајућих парова, поготово у случају машине 3.

	AlexNet	*GoogleNet*	*ResNet-18*	*ResNet-50*
Година настанка	2012	2014	2015	2015
Број слојева	25	144	72	177
Захтевана величина слике	227x227x3	224x224x3	224x224x3	224x224x3
Време „дообучавања“	40 min 11 sec	112 min 53 sec	68 min 13 sec	252 min 15 sec
Просечно време извршавања	0.0772 s	0.2078 s	0.2120	0.3780
Тачност	0.9325	0.9283	0.8687	0.5640

Табела 5 – Резултати „дообучавања“ истренираних модела CNN-а

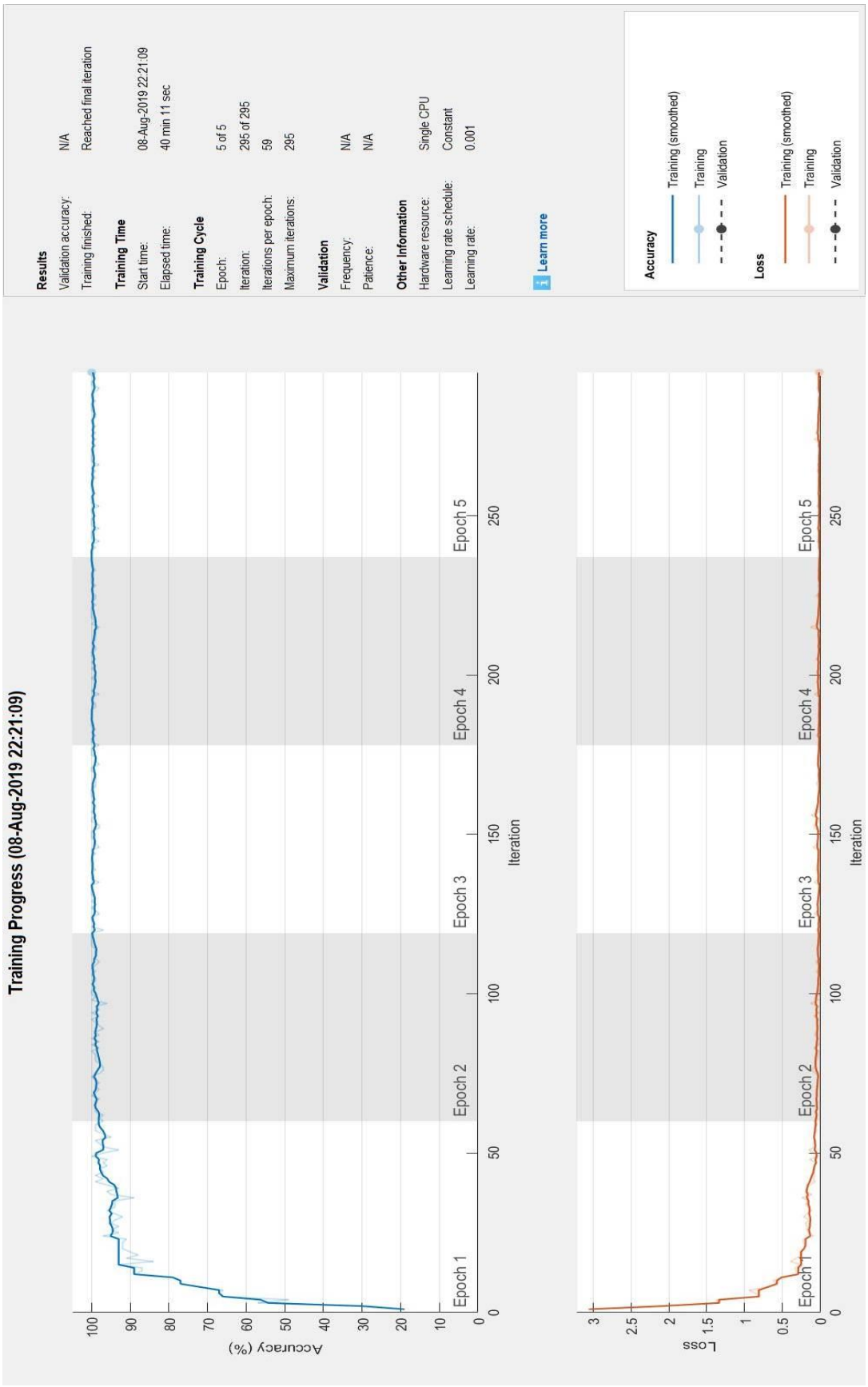


Слика 5.2 – Пример „сличних“ обучавајућих парова машине 2 и машине 3

Како би се уверили у тачност „дообучених“ мрежа, извршено је њихово тестирање на новим скупом слика, при чему су резултати тестирања, заједно са резулатима „дообучавања“, приказани у табели 5.

Из табеле 5 се да приметити да су све мреже, осим *ResNet-50,* уз добре параметре тренирања успеле да постигну високу тачност. Међутим, са порастом броја скривених слојева расте потребно време за „дообучавање“ и извршавање алгоритма. Имајући то у виду, *АлехNet* се показала као најбоља мрежа у оба погледа.

Због напоменутих разлога, за даљу реализацију овог рада користиће се „дообучен“ модел *AlexNet-а*, чији је ток процеса „дообучавања“ представљен на слици 5.3.



Слика 5.3 – Приказ тока процеса „дообучавања“

Оба дијаграма, са слике 5.4, у функцији су броја итерација, при чему је на првом дијаграму приказана конвергенција тачности мреже, а на другом процес минимизације функције циља.

```
Training on single CPU.
Initializing input data normalization.
|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|=========================================================================================|
|       1 |           1 |       00:00:09 |       19.00% |       3.0589 |          0.0010 |
|       1 |          50 |       00:06:06 |       97.00% |       0.0819 |          0.0010 |
|       2 |         100 |       00:13:23 |      100.00% |       0.0088 |          0.0010 |
|       3 |         150 |       00:20:34 |      100.00% |       0.0175 |          0.0010 |
|       4 |         200 |       00:26:47 |       99.00% |       0.0153 |          0.0010 |
|       5 |         250 |       00:32:51 |       99.00% |       0.0151 |          0.0010 |
|       5 |         295 |       00:40:11 |      100.00% |       0.0092 |          0.0010 |
|=========================================================================================|
```

Слика 5.4 – Резултати процеса „дообучавања“ кроз епохе

Са првог дијаграма, као и са слике 5.4, примећује се да мрежа већ у првим итерацијама постиже високу тачност. Након прве епохе брзина конвергенције се постепено смањује, односно тачност мреже почиње да осцилује око једне вредности до краја процеса „дообучавања“. Као што је већ напоменуто, због велике међусобне сличности обучавајућих парова, тачност мреже приликом тестирања, у току процеса обучавања, достиже вредности приближно 100%, што се може видети и на дијаграму. Због тога, мрежа је након „дообучавања“ тестирана на новим скупом слика и као резултат тестирања добила се вредност 93,25% (Табела 5).

На другом дијаграму приказаном на слици 5.3, примећује се да вредност функција циља у првим итерацијама драстично опада, након чега почиње постепено да осцилује око константне вредности. Другим речима са порастом тачности мреже долази до пада функције циља. При тренирању за функцију циља коришћена је функција унакрсне ентропије (енгл. *Cross Entropy Function)*, чија математичка формулација дата једначином 16.

$$H(p,q) = -\sum_{x=1}^{n} p(x) \log q(x) \tag{16}$$

при чему *H* представља излаз из функције, *p* жељени излаз из мреже, *q* стварни излаз из мреже док се са *x* описује детектовани објекат. Посматрајући слику 5.5, поступак израчунавања функције циља за поједине објекте је следећи:

p_1= [0 1 0 0]; q_1= [0.0001 0.9951 0.0000 0.0048];

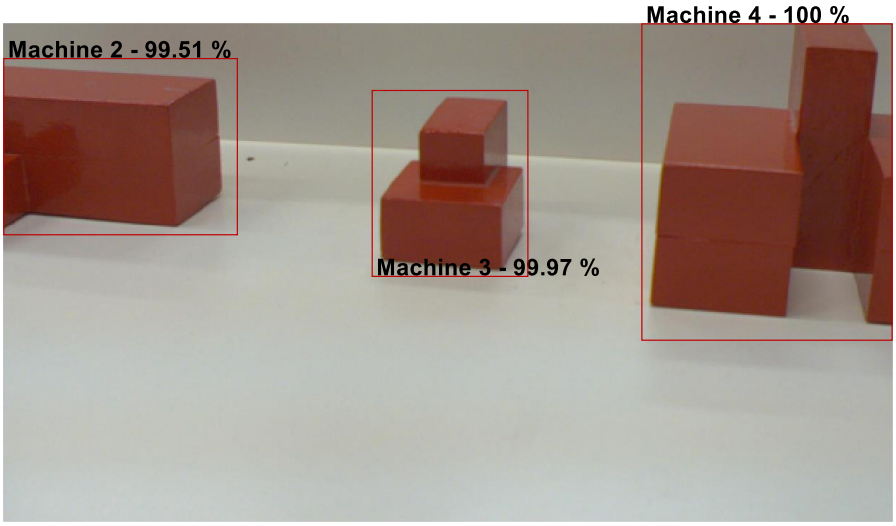
p_2 = [0 0 1 0]; q_2 = [0.0001 0.0000 0.9997 0.0002];

p_3 = [0 0 0 1]; q_3 = [0.0000 0.0000 0.0000 1.0000];

$$H_1 = -0 * \log 0.0001 - 1 * \log 0.9951 - 0 * \log 0.0000 - 0 * \log 0.0048 = -0.0049;$$

$$H_2 = -0 * \log 0.0001 - 0 * \log 0.0000 - 1 * \log 0.9997 - 0 * \log 0.0002 = -0.0003;$$

$$H_3 = -0 * \log 0.0000 - 0 * \log 0.0000 - 0 * \log 0.0000 - 1 * \log 1.0000 = 0;$$



Слика 5.5 – Пример за израчунавање функције циља

Пошто се у овом примеру, за идентификацију објеката користи истрениран модел неуронске мреже, јасно је да ће тачност мреже бити на високом нивоу, а самим тим и вредност функције циља бити веома мала. Међутим, током тренирања мрежа (поготово у првим итерацијама) услед лоше идентификације долази до појаве велике вредности функције циља, на основу које алгоритам учења, након проласка свих слика из дефинисане серије слика, простирањем сигнала уназад (енгл. *back-propagation*) врши промену тежинских односа. Циљ тренирања јесте да се изврши минимизација функције циља, а самим тим и повећа тачност мреже.

Један од начина за визуелно представљање перформанси истрениранoг модела јесте коришћењем матрица одлучивања (слика 5.6). Врсте матрице представљају класу објеката, док колоне представљају остварен излаз из мреже.

На дијагонали матрице се налази број тачно идентификованих објеката, док све остале вредности представљају лоше идентификоване објекте. Узимајући машину 3 за пример, од укупно 254 слике објекта, алгоритам је 4 пута проценио да је упитању машина 1, 10 пута машина 2 и 240 пута жељена машина 3. Иста аналогија важи за остале машине при чему се види да алгоритам, због сложености облика, највише греши приликом идентификације машине 4.

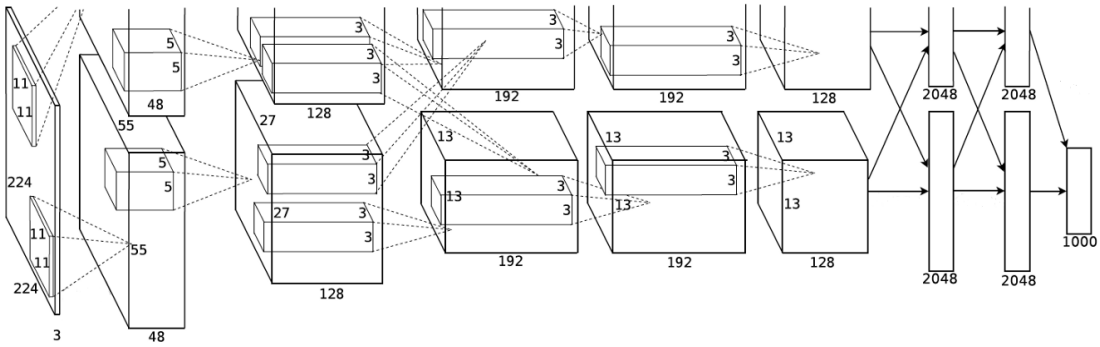
True Class	Machine 1	Machine 2	Machine 3	Machine 4
Machine 1	120	2	4	4
Machine 2	3	277	10	16
Machine 3	1	1	240	10
Machine 4	4	1		213

Слика 5.6 – Матрица одлучивања

5.2 AlexNet

Својим радом [17], Алекс Крижевски (*Alex Krizhevsky*) представио је нови приступ у решавању проблема класификације, и тиме револуционирао домен конволуционих неуронских мрежа. У том раду, представљен је истрениран модел конволуционе неуронске мреже под називом *AlexNet*. На такмичењу ILSVRC (*ImageNet Large Scale Visual Recognition Challenge*) 2012. године, *AlexNet* је остварила победу, постижући „*top 5 test error*“ од 15.4% (10.8% мање од другопласираног).

Мрежа је тренирана 6 дана на преко милион слика, са *ImageNet* базе података [21], са SGDM алгоритмом обучавања, да класификује 1000 класа објеката, као што су нпр.: оловка, брод, тастатура, мачка, лопта итд. Архитектура мреже (слика 5.7) подсећа на арихитектуру *LeNet*-а [22], при чему је основна разлика у количини слојева и броју конволуционих филтера по слоју.



Слика 5.7 – Упрошћен приказ архитектуре AlexNet-а [17]

На слици 5.8, приказани су сви слојеви *AlexNet-а,* у које се убрајају: конволуциони слојеви (са величином филтера 11x11,5x5 и 3x3 пиксела), слојеви максималног сажимања, потпуно повезани слојеви, изостављајући слојеви, слојеви нормализације и слојеви активационих функција (*ReLU* и *Softmax*).

1	'data'	Image Input	227x227x3 images with 'zerocenter' normalization
2	'conv1'	Convolution	96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]
3	'relu1'	ReLU	ReLU
4	'norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
5	'pool1'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
6	'conv2'	Grouped Convolution	2 groups of 128 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]
7	'relu2'	ReLU	ReLU
8	'norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
9	'pool2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
10	'conv3'	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
11	'relu3'	ReLU	ReLU
12	'conv4'	Grouped Convolution	2 groups of 192 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
13	'relu4'	ReLU	ReLU
14	'conv5'	Grouped Convolution	2 groups of 128 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
15	'relu5'	ReLU	ReLU
16	'pool5'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
17	'fc6'	Fully Connected	4096 fully connected layer
18	'relu6'	ReLU	ReLU
19	'drop6'	Dropout	50% dropout
20	'fc7'	Fully Connected	4096 fully connected layer
21	'relu7'	ReLU	ReLU
22	'drop7'	Dropout	50% dropout
23	'fc8'	Fully Connected	1000 fully connected layer
24	'prob'	Softmax	softmax
25	'output'	Classification Output	crossentropyex with 'tench' and 999 other classes

Слика 5.8 – Списак слојева архитектуре AlexNet-а

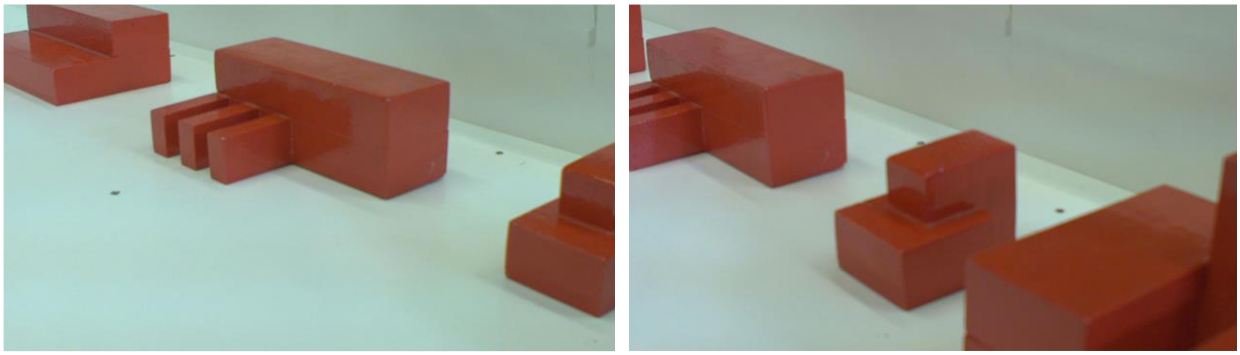
6.0 Модул за идентификацију детектованих ликова објеката

У оквиру овог поглавља, биће приказани и детаљно описани кораци приликом идентификације технолошких ентитета са слике уз коришћење алгоритма за детекцију ликова објеката, објашњеног у поглављу 4. Целокупан алгоритам, по коме се извршава идентификације технолошких ентитета, приказан је на слици 6.1.

Слика 6.1 – Алгоритам за идентификацију технолошких ентитета

1. Учитавање слика и CNN-а

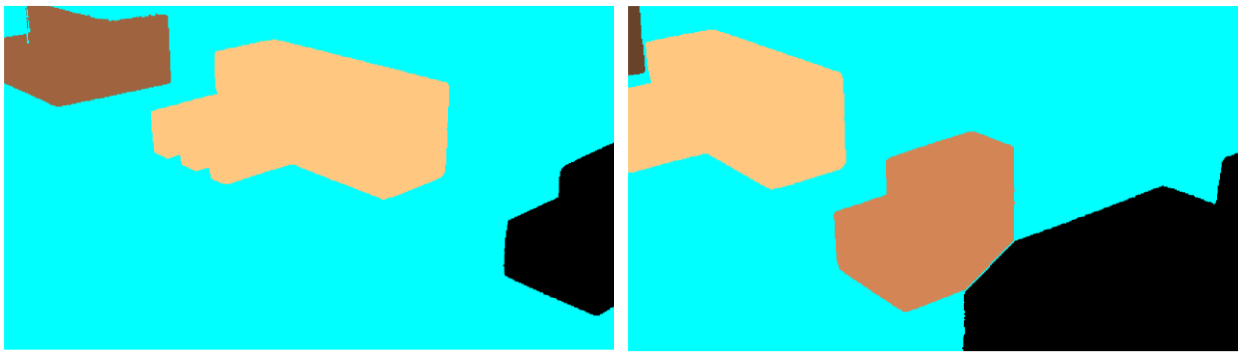
Аналогно процедури описаној у поглављу 4, извршава се аквизиција слика са леве и десне камере (слика 6.2), након чега следи учитавање одабраног модела CNN-а.



Слика 6.2 – Слике добијене аквизицијом података са леве и десне камере

2. Сегментација учитаних слика

Овај корак је детаљно елабориран у поглављу 4.0, при чему су резултати примене алгоритма приказани на слици 6.3.

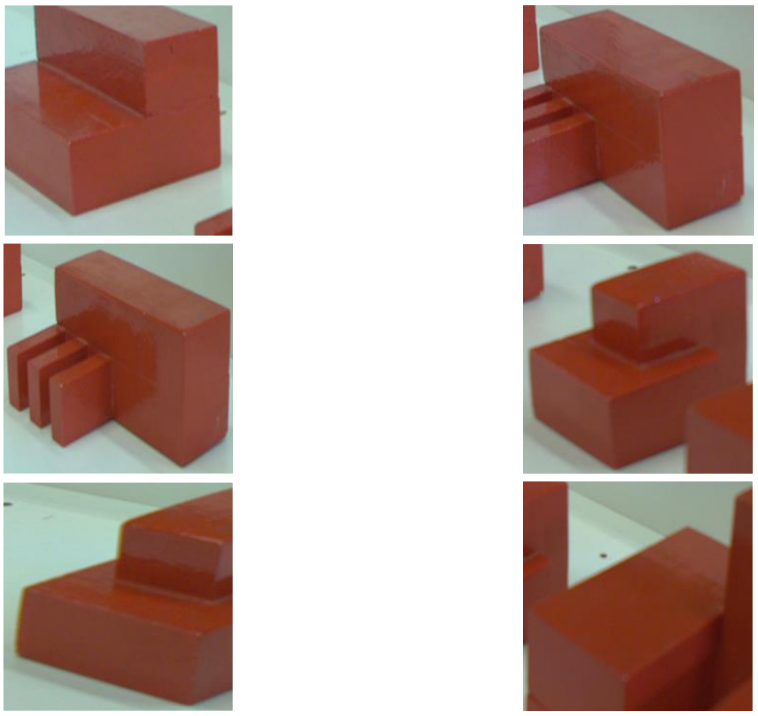


Слика 6.3 – Резултати сегментације учитаних слика

3. Екстракција слика детектованих ликова објеката

У овом кораку извршава се формирање и екстракција појединачних слика детектованих ликова објеката како би у наредном кораку, на основу њих, могли да се идентификују. Екстракције слика је базирана на проналажење оквира региона (енгл. *Bounding Box),* које се формирају око спољашње контуре детектованих објеката. Пошто се приликом коришћења CNN-а захтева одређена резолуција слика, која код *AlexNet*-а износи 227x227x3, пре прослеђивања екстракованих слика мрежи изврши модификације њихових резолуција.

На слици 6.4, су представљене добијене појединачне слике свих детектованих ликова објеката са леве и десне камере.



Слика 6.4 – Појединачне слике детекованих ликоба објеката са леве и десне камере

4. Идентификација детектованих ликова објеката

Према поглављу 5.1, експерименталним испитивањима се доказало да је за идентификацију детектованих ликова објеката најбоље резултате показао „дообучен“ модел *AlexNet*-а, па се та мрежа користила у даљој реализацији рада. Улаз у мрежу представљају слике добијене кораком 3 (слика 6.4), а као излаз из мреже добија се вектор који представља дистрибуцију вероватноће потенцијалних исхода. Пошто у постављеном проблему постоје четири могућа исхода за детектовани лик објекта, вектор дистрибуције вероватноће је величине 1x4. Вредности унутар вектора представљају вероватноћу детектоване класе, при чему се класа са највећом вероватноћом бира као излаз из мреже, односно као идентификација детектованог лика објекта.

Добијени вектори дистрибуције вероватноће су, по редоследу са слике 6.4, представљени у табели 6. На основу добијене табеле, примећује се да је нпр. за прву детектовану машину на десној слици, алгоритам је на излазу генерисао вероватноћу од 36.73% да је на слици приказана машина 1, 60,87% машина 2, 0.7% машина 3 и 1.7% машина 4.

$score_1 = [0.9953 \quad 0.0045 \quad 0.0002 \quad 0.0001]$	$score_1 = [0.3673 \quad 0.6087 \quad 0.0070 \quad 0.0170]$
$score_2 = [0.0160 \quad 0.9822 \quad 0.0000 \quad 0.0018]$	$score_2 = [0.0084 \quad 0.0156 \quad 0.9083 \quad 0.0677]$
$score_3 = [0.0210 \quad 0.0086 \quad 0.8625 \quad 0.1079]$	$score_3 = [0.0060 \quad 0.0004 \quad 0.0002 \quad 0.9934]$

Табела 6 – Вектори дистрибуције вероватноће потенцијалних класа

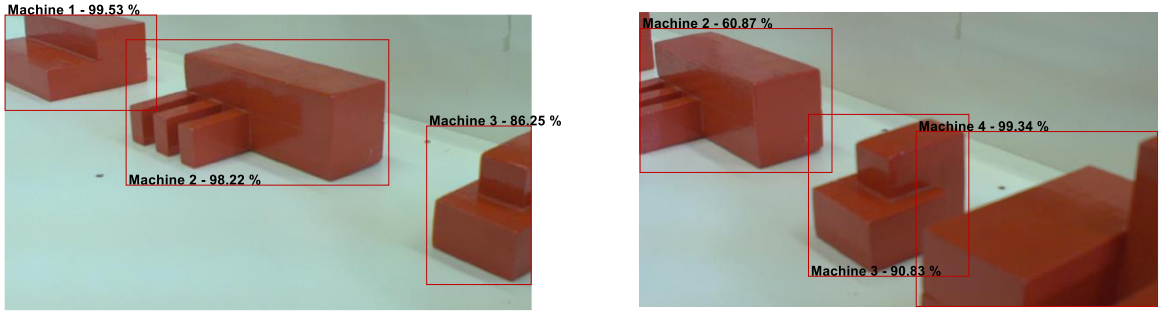
5. Конектовање ликова објеката истих идентификованих класа

Након идентификације свих детектованих ликова објеката на слици, потребно је да се провери да ли неки од детектованих ликова припадају истом објекту, односно да ли је „Вододелница“ алгоритам направио грешку приликом сепарације преклапајућих објеката. Као што је објашњено у поглављу 3.6, уколико се на слици примети да постоје више објеката са истом идентификацијом, који се налазе у непосредној близини један другом, извршава се њихово спајање. Након спајања, поново се врши идентификација објеката на слици, при чему се првобитно раздвојени објекти сада посматрају као један. Успешност овог корака чврсто зависи од способности мреже да тачно идентификује подељене делове истог објекта.

Пошто у овом примеру „Вододелница“ алгоритам није направио грешку тј. није било објеката са истом идентификацијом, идентификација објеката је остала непромењена.

6. Репрезентације идентификованих објеката

Након извршене идентификације, и провере „Вододелница“ алгоритма, преостало је да се на основу табеле 6 и оквира региона на адекватан начин представе постигнути резултати идентификације технолошких ентитета (слика 6.5). Уколико се на обе слике детектују исти технолошки ентитети, могуће је извршити упаривање њихових карактеристичних објеката чиме се омогућава управљање кретањем интелигентног мобилног робота RAICO у лабораторијском моделу технолошког окружења.



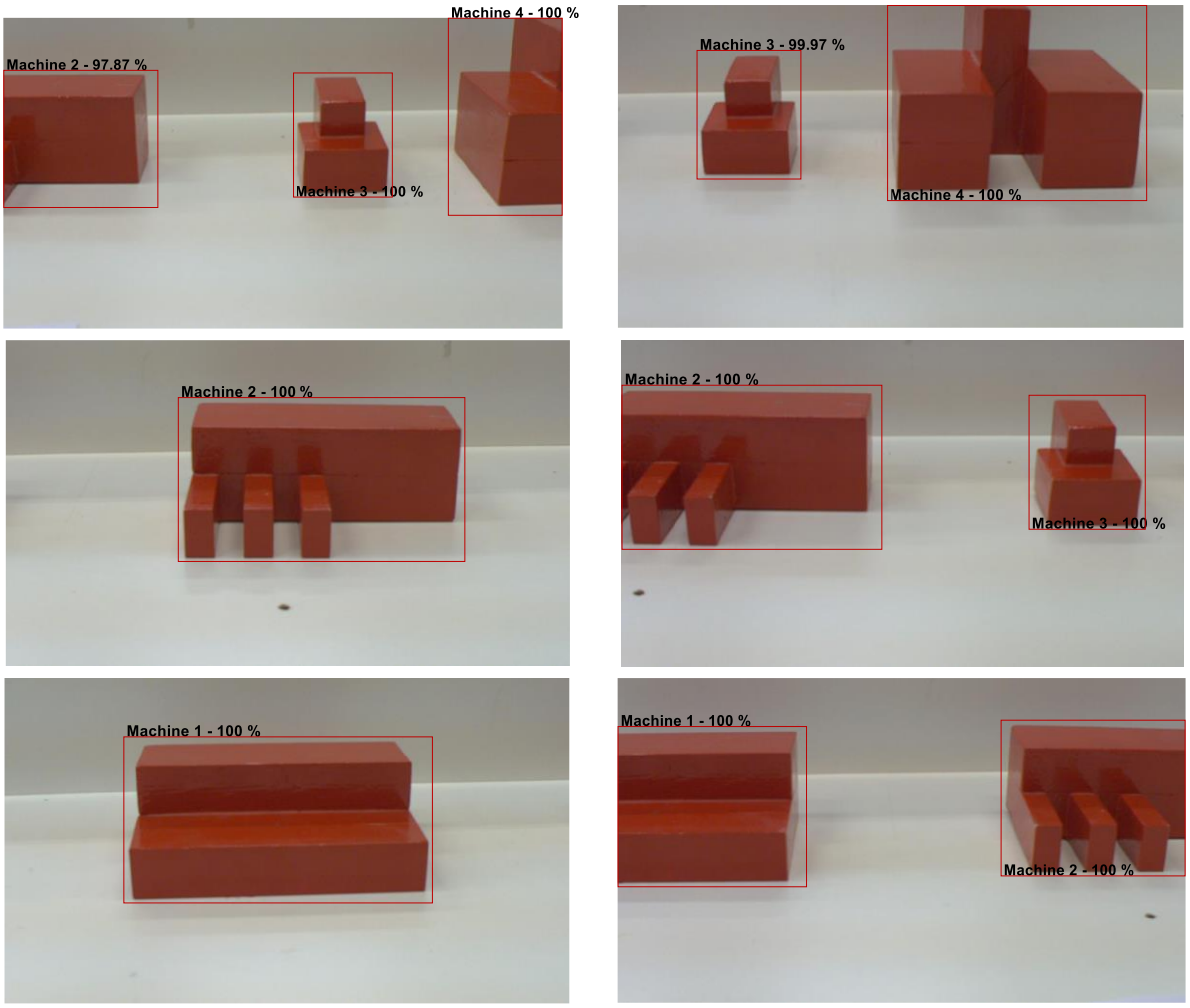
Слика 6.5 – Резултати идентификације технолошких ентитета

7.0 Експериментална верификација развијеног алгоритма

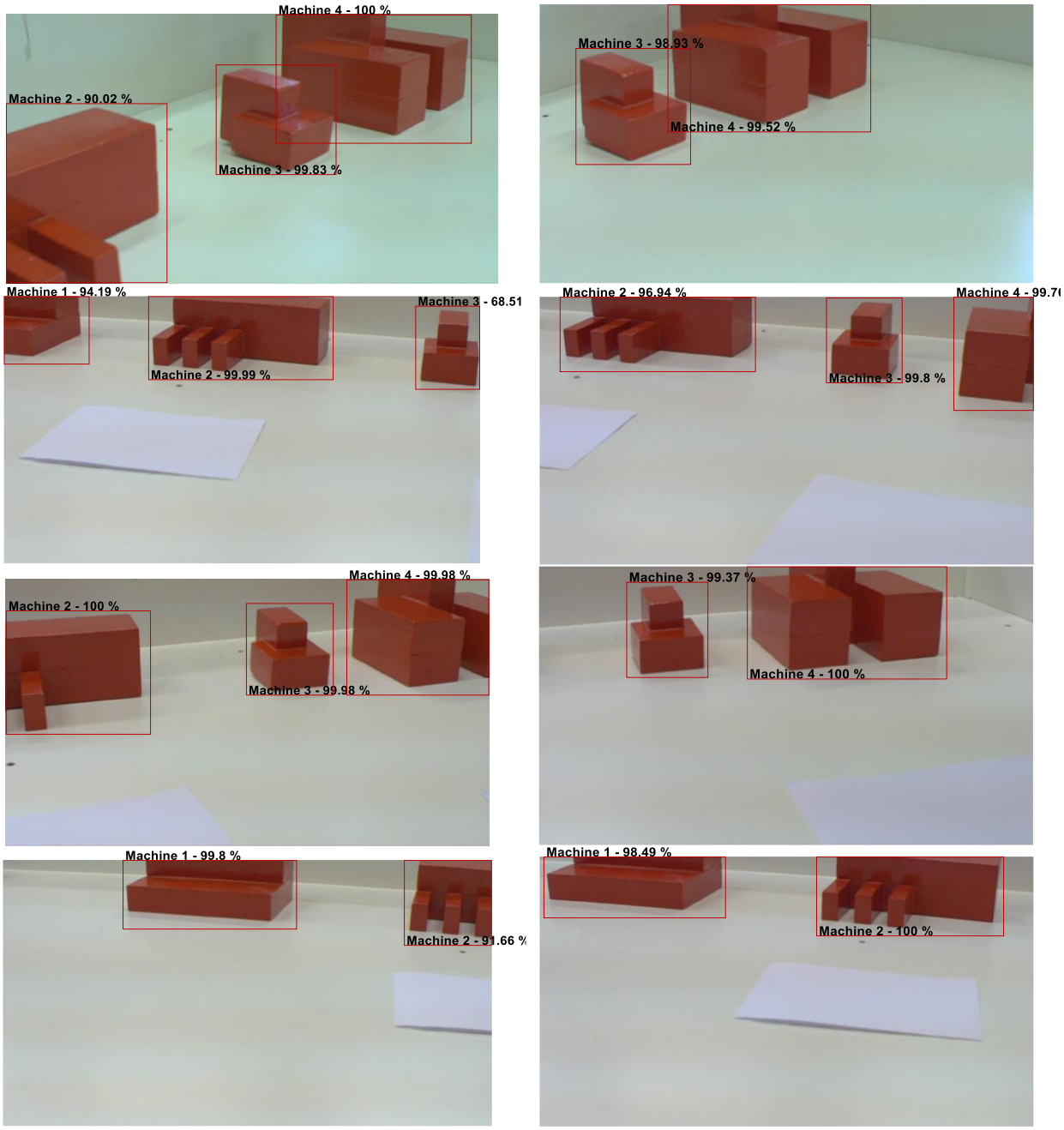
7.1 Примена и анализа тачности развијеног модула на реалним сликама лабораторијског модела технолошког окружења

У овом поглављу су представљени резултати примене развијеног модула за идентификацију технолошких ентитета на реалним сликама лабораторијског модела технолошког окружења, добијене коришћењем интелигентног мобилног робота RAICO са две Basler acA1920-25uc камера. У првој фази, приказани су случаји на којима је модул тачно извршио идентификацију свих технолошких ентитета, при чему су лево приказане слике добијене са леве камере, а десно слике добијене са десне камере. Након тога следи приказ и образложење случаја нетачне идентификације објеката.

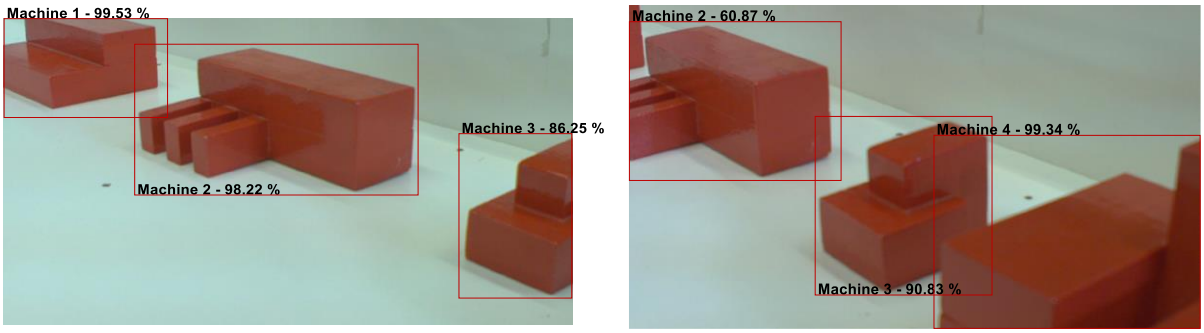
На сликама 7.1 ,7.2 и 7.3, приказане су сцене добијене из различитих оријентација и позиција мобилног робота, при чему упркос комплексним трансформацијама које ликови објеката доживљавају приликом промене положаја мобилног робота, алгоритам успева да правилно изврши идентификацију технолошких ентитета.



Слика 7.1 – Примери тачне идентификације технолошких ентитета, 1



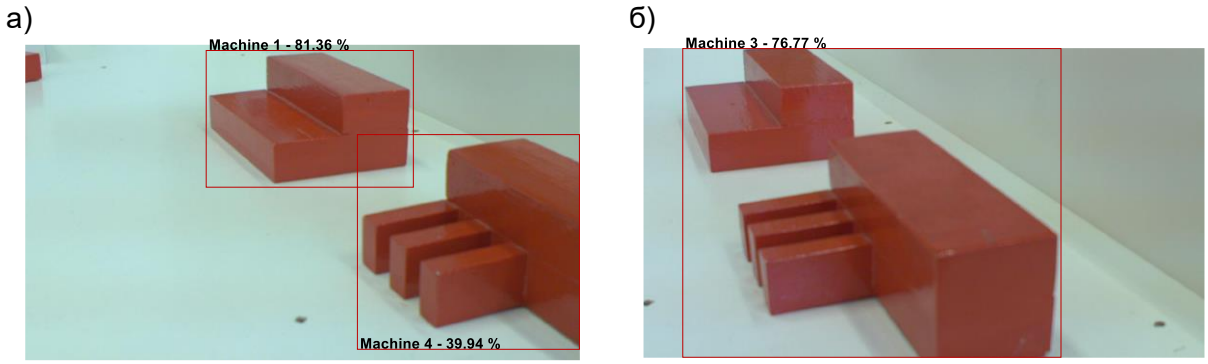
Слика 7.2 – Примери тачне идентификације технолошких ентитета, 2



Слика 7.3 – Примери тачне идентификације технолошких ентитета, 3

На слици 7.4 а), иако је сегментација објеката коректно одрађена, CNN није успела да адекватно идентификује машину 2. На основу добијеног вектора дистрибуције вероватноћа $score_2 = [0.0108 \quad 0.2822 \quad 0.3076 \quad 0.3994]$, примећује се да су вероватноће идентификације машина 2, 3 и 4 приближно исте, што значи да идентификација није адекватно извршена.

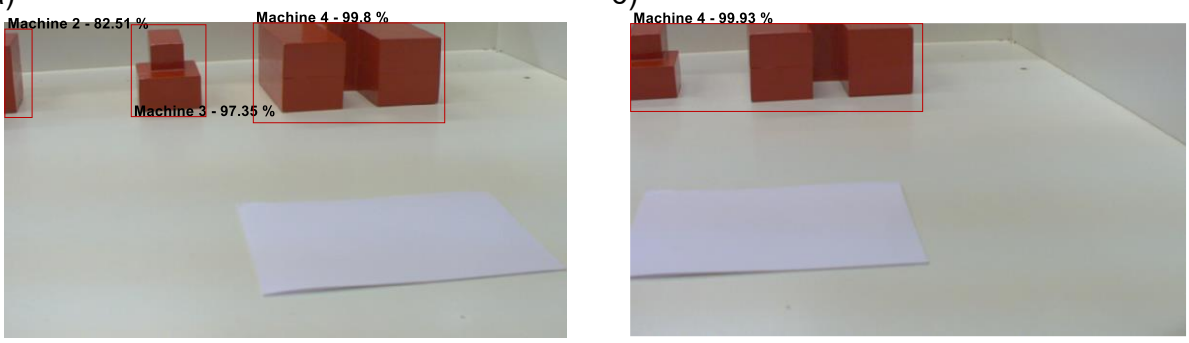
На слици 7.4 б), након добре сепарације објеката „Вододелница“ алгоритмом, дошло је до њихове лоше идентификације. Мрежа је у оба случаја на свом излазу идентификовала машину 4, тако да се у кораку корекције „Вододелница“ алгоритма извршило њихово спајање. Затим је уследила лоша идентификација спојених објеката, генеришући на илазу машину 3.



Слика 7.4 – Случај нетачно идентификованих технолошких ентитета, пример 1

На слици 7.5 а) је тачно одрађена идентификација технолошких ентитета док је на слици 7.5 б), након тачне сегментације детектованих ликова објеката, алгоритам прави грешку услед лоше идентификације машине 3, генеришући на излазу машину 4. Грешку је у овом случају проузроковала чињеница да „исечен“ део машине 3 подсећа на умањен приказ десног дела машине 4. Пошто се на слици налазе две машине са истом идентификационом ознаком, у кораку корекције „Вододелница“ алгоритма долази до њиховог спајања, након чега због појаве доста карактеристика машине 4 унутар оквира региона, мрежа са великом сигурношћу тврди да се ради управо о тој машини.

а) б)

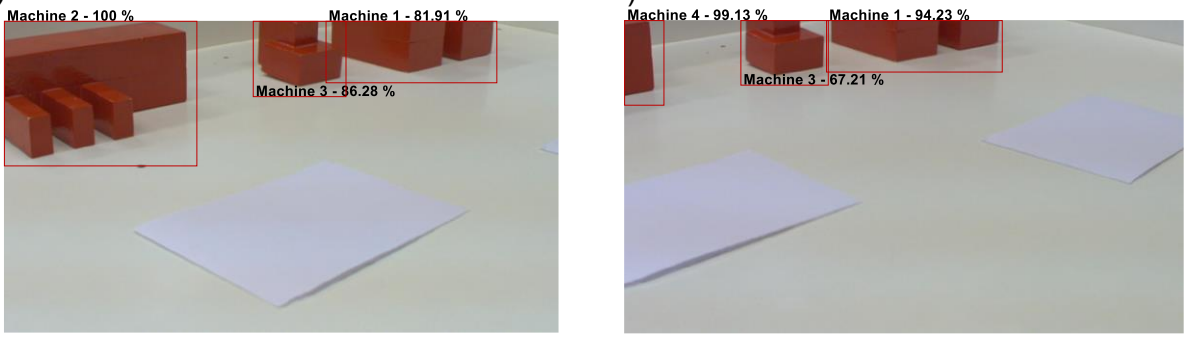


Слика 7.5 – Случај нетачно идентификованих технолошких ентитета, пример 2

На слици 7.6 а), алгоритам је након тачне детекције и раздвајања ликова објеката направио грешку приликом идентификације машине 4. Грешка настаје услед недовољне приказаности машине, што представља неизбежну ситуацију приликом кретања интелигентног мобилног робота кроз лабораторијски модел технолошког окружења. Са друге стране, не може се ни очекивати од мреже да на основу детекције тако малог дела објекта направи добру естимацију класе.

На слици 7.6 б), аналогно претходном примеру, грешка настаје услед детекције малог дела машине 2 и машине 4.

а) б)

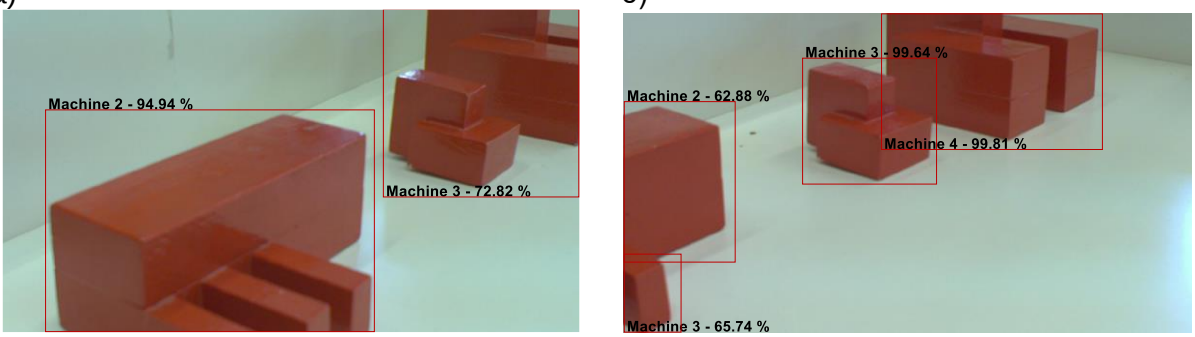


Слика 7.6 – Случај нетачно идентификованих технолошких ентитета, пример 3

На слици 7.7 а), грешка настаје приликом извршавања „Вододелница“ алгоритма. Како је преклапање машине 3 и машине 4 велико, алгоритам не препознаје да су то два различита објекта, па се самим тим сепарација не извршава. Приликом идентификације детектованог региона због целокупног приказа машине 3, алгоритам са великом сигурношћу додељује целом региону идентификацију машине 3.

На слици 7.7 б) се дешава супротна ситуација: „Вододелница“ алгоритам на машини 2 препознаје два локална минимума, односно две неповезане групе пиксела које имају велику Еуклидску удаљеност од околине, и на основну њих врши сепарацију машину 2 на два дела.

а) б)



Слика 7.7 – Случај нетачно идентификованих технолошких ентитета, пример 4

7.2 Експеримент

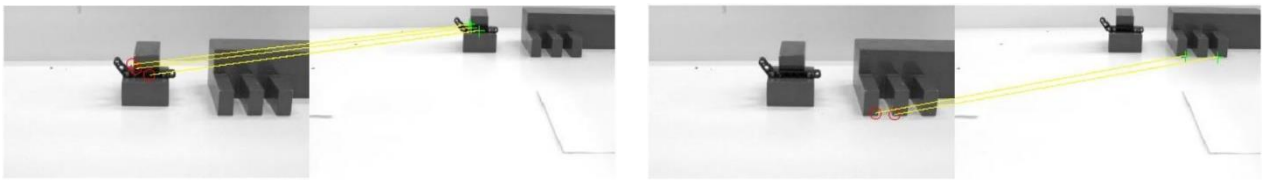
Како би се проверила валидност развијеног алгоритма, извршено је експериментално испитивање коришћењем интелигентног мобилног робота RAICO, на коме су уграђене две Basler acA1920-25uc камере.

Експеримент се започиње тако што се мобилном роботу представе две слике (једна за леву камеру и једна за десну камеру) технолошког окружења, добијене из положаја у ком робот треба да се нађе. Упаривањем карактеристичних објеката (енгл. *Point Features)*, односно карактеристичних тачака на детектованим машинама, са тренутних и циљаних слика, извршава се управљање кретањем интелигентног мобилног робота кроз лабораторијски модел технолошког окружења. Коришћењем алгоритма за идентификацију, знатно се побољшало проналажење карактеристичних објеката, а тиме и њихово упаривање. Експеримент се завршава када поклапање карактеристичних објеката достигне задовољавајући ниво.

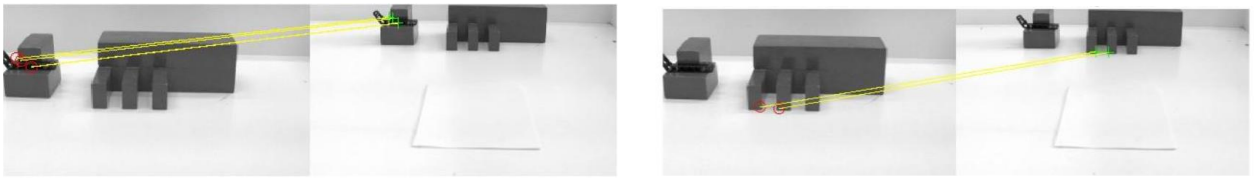
У наставку биће приказани детектоване машине и упарени карактеристични објекти, на левој и десној слици, свих секвенцијалних корака кретања мобилног робота кроз лабораторијски модел технолошког окружења. На сликама 7.8 – 7.35 су следствено приказане упарене машине са леве и десне слике, а затим и детектовани карактеристични објекти на левој и десној слици.

Корак 1:

Слика 7.8 – Детектоване машине са леве и десне камере, корак 1



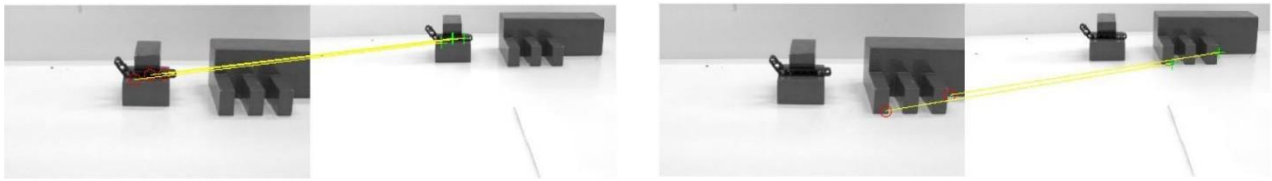
Слика 7.9 – Правилно упарени карактеристични објекти на левој камери, корак 1



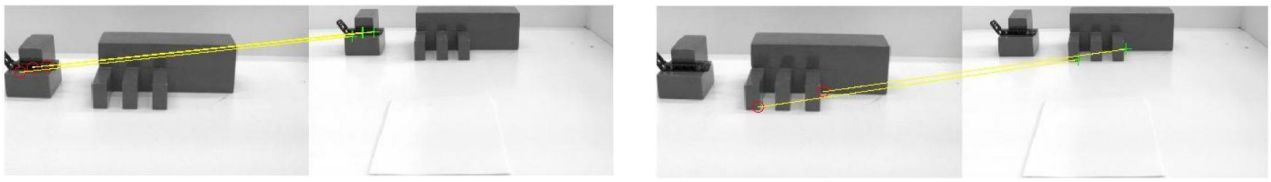
Слика 7.10 – Правилно упарени карактеристични објекти на десној камери, корак 1

Корак 2:

Слика 7.11 – Детектоване машине са леве и десне камере, корак 2



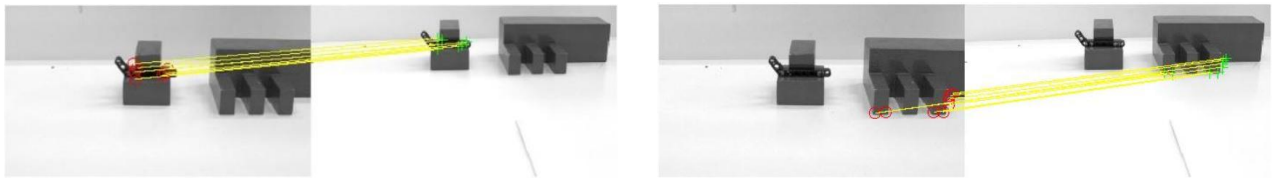
Слика 7.12 – Правилно упарени карактеристични објекти на левој камери, корак 2



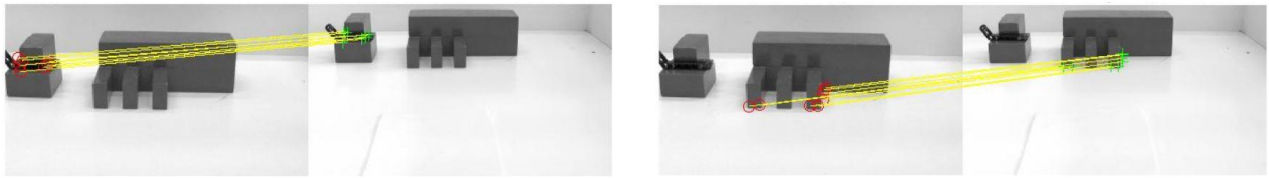
Слика 7.13 – Правилно упарени карактеристични објекти на десној камери, корак 2

Корак 3:

Слика 7.14 – Детектоване машине са леве и десне камере, корак 3



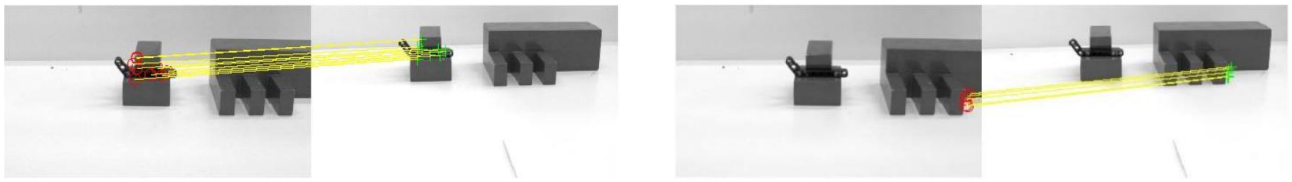
Слика 7.15 – Правилно упарени карактеристични објекти на левој камери, корак 3



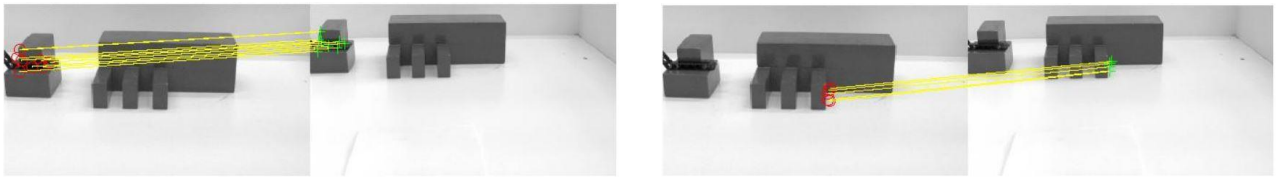
Слика 7.16 – Правилно упарени карактеристични објекти на десној камери, корак 3

Корак 4:

Слика 7.17 – Детектоване машине са леве и десне камере, корак 4



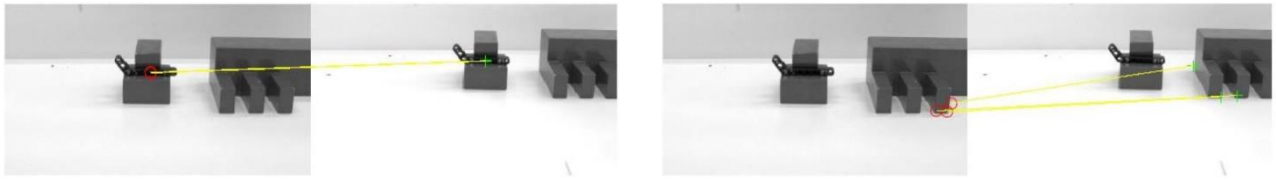
Слика 7.18 – Правилно упарени карактеристични објекти на левој камери, корак 4



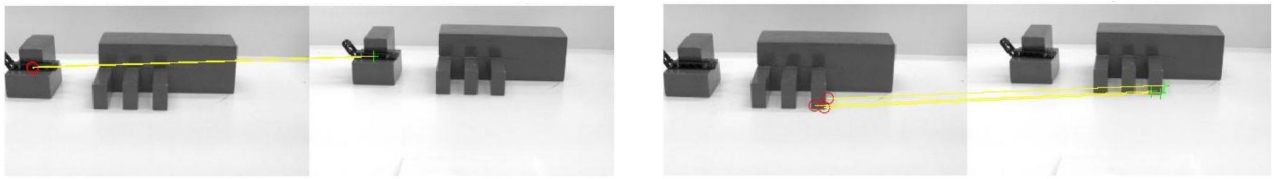
Слика 7.19 – Правилно упарени карактеристични објекти на десној камери, корак 4

Корак 5:

Слика 7.20 – Детектоване машине са леве и десне камере, корак 5



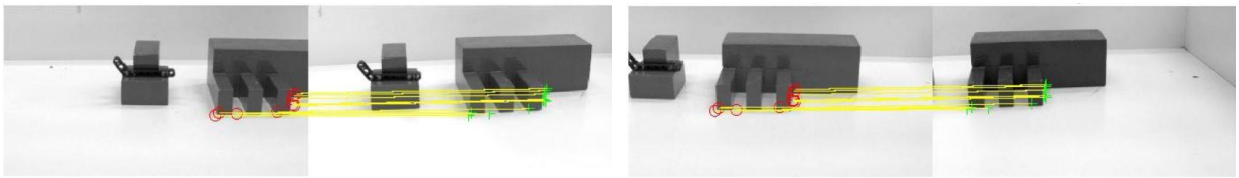
Слика 7.21 – Правилно упарени карактеристични објекти на левој камери, корак 5



Слика 7.22 – Правилно упарени карактеристични објекти на десној камери, корак 5

Корак 6:

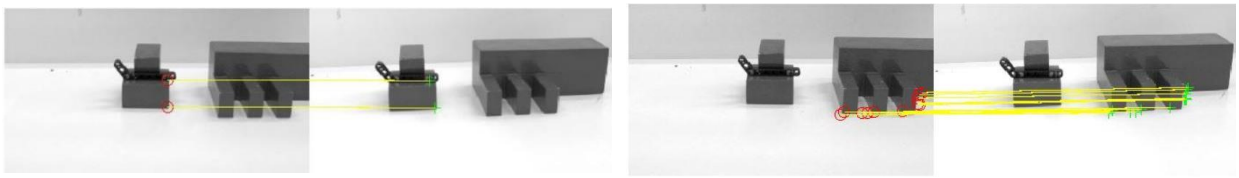
Слика 7.23 – Детектоване машине са леве и десне камере, корак 6



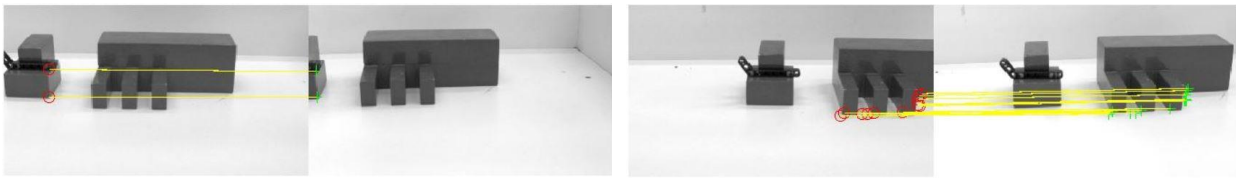
Слика 7.24 – Правилно упарени карактеристични објекти на левој и десној камери, корак 6

Корак 7:

Слика 7.25 – Детектоване машине са леве и десне камере, корак 7



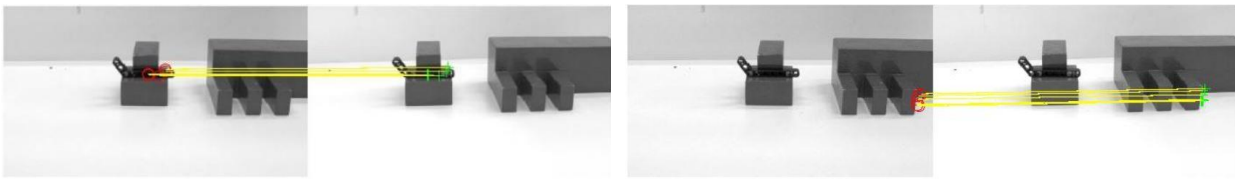
Слика 7.26 – Правилно упарени карактеристични објекти на левој камери, корак 7



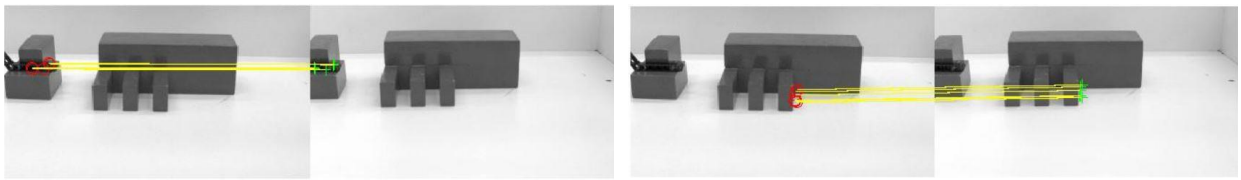
Слика 7.27 – Правилно упарени карактеристични објекти на десној камери, корак 7

Корак 8:

Слика 7.28 – Детектоване машине са леве и десне камере, корак 8



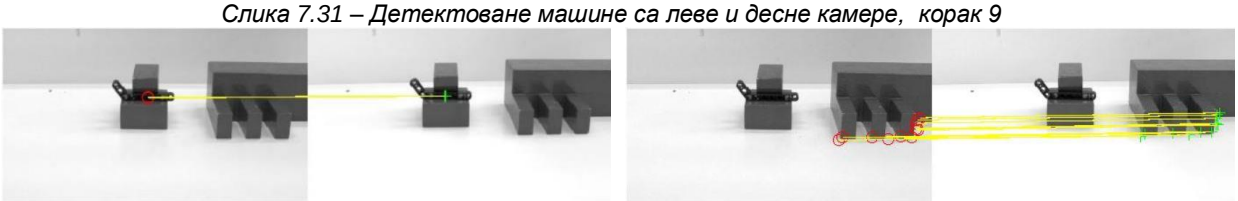
Слика 7.29 – Правилно упарени карактеристични објекти на левој камери, корак 8



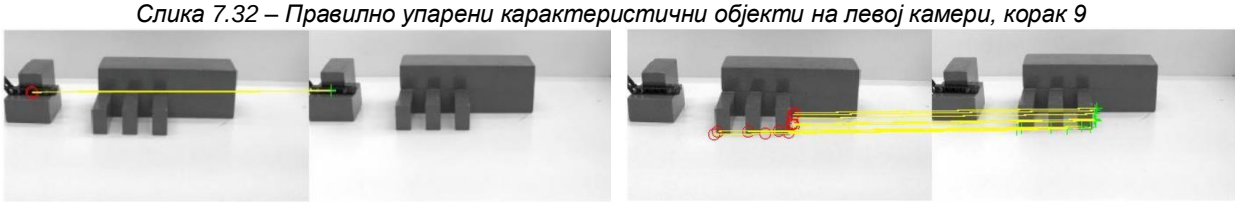
Слика 7.30 – Правилно упарени карактеристични објекти на десној камери, корак 8

Корак 9:

Слика 7.31 – Детектоване машине са леве и десне камере, корак 9



Слика 7.32 – Правилно упарени карактеристични објекти на левој камери, корак 9



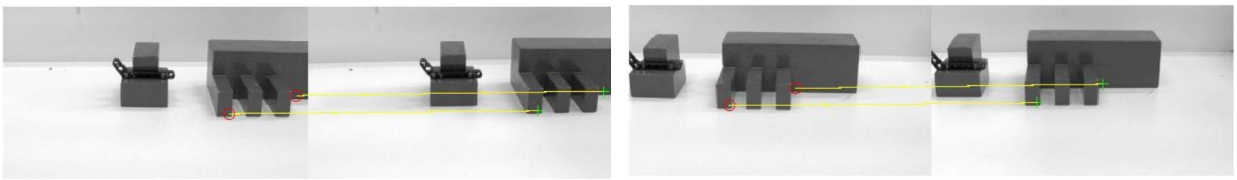
Слика 7.33 – Правилно упарени карактеристични објекти на десној камери, корак 9

Приликом извођења приказаног експеримента, алгоритам је са великом тачношћу успео да идентификује технолошке ентитета на представљеним сликама, доказујући да се коришћењем алгоритма за идентификацију технолошких ентитета побољшава управљање кретањем интелигентног мобилног робота RAICO. Међутим, пошто је алгоритам у сваком кораку успешно детектовао приказане технолошке ентитете, овим се не може приказати реална слика функционисања алгоритма. Приликом извођења експеримената, грешке су проузроковане услед:

- лоше идентификације детектованих ликова објеката,
- неуспешне сепарације преклапајућих машина,
- лоше сепарације преклапајућих објеката на слици,
- конектовање погрешних ликова објеката.

У наставку, на сликама 7.34 и 7.35, биће приказан пример једног корака где се лоша сегментација извршила услед конектовања погрешних ликова објеката.

Слика 7.34 – Детектоване машине са леве и десне камере



Слика 7.35 – Правилно упарени карактеристични објекти на левој и десној камери

На основу извршених тестирања, закључује се да грешке, у већини случаја, настају услед немогућности мреже да тачно идентификује детектоване ликове објекта. Поред тога, грешка у појединим случајевима је проузрокована приликом сепарације објеката, јер се применом „Вододелница“ алгоритма не може превише утицати на положај самих „вододелница“.

Алгоритам је тестиран на укупно 400 слика при чему је успешно успео да идентификује 373 слика, што значи да проценат тачности целог алгоритма износи 93.25%. Како су слике на којима је било извршено тестирање генерисане у различитим условима, може се сматрати да добијена тачност представља реалну слику функционисања алгоритма.

8. Закључак

У овом мастер раду, развијен је модификовани алгоритам за идентификацију технолошких ентитета, базиран на коришћењу сегментације слике и конволуционих неуронских мрежа. Алгоритам је реализован програмирањем у *MATLAB* софтверском пакету, а састоји се из две подцелине: алгоритма за детекцију ликова објеката и алгоритма за идентификацију технолошких ентитета.

Приликом детекције ликова објеката, извршене су бројне трансформације над учитаном сликом, у које се убрајају: „замућење“ слике, уз примену Гаусовог филтера, проналажење ивица објеката на слици применом методе сегментације базиране на детекцији ивица, груписање пиксела у регионе и сепарација преклапајућих објеката слике коришћењем „Вододелница“ алгоритма.

Како би детекција ивица била на потребном нивоу, извршена је компаративна анализа доступних метода сегментације базираних на детекцији ивица објеката, при чему се Собелова метода показала као оптимално решење за постављени проблем. Након груписања пиксела у регионе, на основу анализе слике објеката, постојећим методама за репрезентацију слика у боји, утврђено је који региони припадају објектима, а који окружењу. Применом „Вододелница“ алгоритама извршена је сепарација преклапајућих објеката, и формирање коначне слике детектованих ликова објеката.

На основу слике детектованих ликова објеката, екстракцијом засебних слика технолошких ентитета, формирани су обучавајући парови. Процесирањем екстракованих слика технолиошких ентитета применом конволуционих неуронских мрежа изршена је њихова идентификација, при чему је резолуција сваке екстраковане слике подешена да одговара захтеваној резолуцији слике коју мрежа користи (227x227 пиксела). Тестирањем обучених модела конволуционих неуронских мрежа, закључило се да је „дообучен“ модел *AlexNet-a,* уз најкраће време извршавања, постигао најбоље резултате у погледу успешне идентификације технолошких ентитета. Уколико је дошло до сепарације делова исте машине услед примене „Вододелница“ алгоритма, на основу идентификације детектованих региона одређене тачности, извршиће се њихово спајање. Идентификацијом приказаних технолошких ентитета унапређује се упаривање карактеристичних објеката (енгл. *Features*) на основу којих се извршава интелигентно управљање мобилног робота у оквиру постојећег лабораторијског модела технолошког окружења.

Валидација алгоритма за идентификацију технолошких ентитета, извршена је коришћењем интелигентног мобилног робота RAICO, са уграђеним двема Basler acA1920-25uc камерама. Приликом идентификације 400 слика објекта у оквиру сцене, алгоритам је са тачношћу од 93.25% успешно идентификовао приказане технолошке ентитете. На основу добијених резултата, може да се закључи да је алгоритам за идентификацију технолошких ентитета успешно примењен, првенствено током извођења експеримената, али и да је подложан грешкама уколико услови у окружењу нису адекватни, попут осветљења објеката у оквиру сцене. Грешке, уочене приликом идентификације технолошких ентитета, претежно су изазване променом нивоа осветљености унутар технолошког окружења, као и променом нивоа рефлектоване светлости. CNN је показала задољавајуће резултате приликом идентификације, при чему су грешке настајале приликом идентификације недовољно видљивих ликова објеката. Идентификација технолошких ентитета може да се побољша тренирањем мреже са више прикупљених репрезентативних обучавајућих парова.

9. Литература

[1] Миљковић З., Александрић Д., Вештачке неуронске мреже, Универзитет у Београду – Машински факултет, Београд, 2009.

[2] Миљковић З., Петровић М., Изводи са предавања на предмету Интелигентни технолошки системи, Универзитет у Београду – Машински факултет, Београд 2019

[3] Миљковић З.: Системи вештачких неуронских мрежа у производним технологијама, Машински факултет Београд, Монографија, 2003.

[4] Peter C., Robotics, Vision and Control Fundamental Algorithms in MATLAB, Springer Tracts in Advanced Robotics Volume 73, 2017

[5] http://what-when-how.com/introduction-to-video-and-image-processing/neighborhood-processing-introduction-to-video-and-image-processing-part-1/, - датум последњег приступа 23.08.2019

[6] https://en.wikipedia.org/wiki/Object_detection - датум последњег приступа 23.08.2019

[7] https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5 - датум последњег приступа 23.08.2019

[8] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 - датум последњег приступа 23.08.2019

[9] https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5 - датум последњег приступа 23.08.2019

[10] Linda G. Shapiro, George C. Stockman, “Computer Vision”, ISBN 0-13-030796-3, New Jersey, Prentice-Hall, , 2001

[11] https://www.techopedia.com/definition/190/artificial-intelligence-ai - датум последњег приступа 23.08.2019

[12] Шокац М., Будак И., Јаковљевић Ж., Сантошти Ж., Вукелић Ђ., Преглед стања и анализа најћешће примењиваних метода за сегментацију слике, *International Scientific Conference – ETIKUM*, Нови Сад 2018

[13] people.vts.su.ac.rs/~pmiki/STOREnGO/DIP/Poglavlje%209%20-%20Segmentacija%20slike%20(Short).pdf - датум последњег приступа 01.09.2019

[14] https://en.wikipedia.org/wiki/RGB_color_model **-** датум последњег приступа 23.08.2019

[15] https://en.wikipedia.org/wiki/HSL_and_HSV - датум последњег приступа 23.08.2019

[16] https://www.mathworks.com/company/newsletters/articles/the-watershed-transform-strategies-for-image-segmentation.html - датум последњег приступа 23.08.2019

[17] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." *Advances in neural information processing systems*., Toronto, 2012.

[18] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In *Proceedings of the IEEE conference on computer vision and attern recognition*, pp. 1-9., Toronto 2015.

[19] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778., Toronto 2016

[20] https://www.mathworks.com/help/deeplearning/gs/get-started-with-transfer-learning.html - датум последњег приступа 23.08.2019

[21] http://www.image-net.org - датум последњег приступа 23.08.2019

[22] LeNet - Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik Bell Laboratories, Holmdel, NJ 07733, USA, 2014

[23] Haralick R.M., Shapiro L.G., Computer and Robot Vision, Volume I, II, Addison – Wesley Publishing Company, 1992.

[24] Szeliski R., Computer Vision: Algorithm and Application, Springer Science & Business Media, 2010.

[25] Миљковић З., Славковић Н., Изводи са предавања из предмета Индустријски роботи, Универзитет у Београду – Машински факултет, Београд, 2018

[26] Вујић Н., Јовановић С., Милетић Т.,Јованчићевић Л., Живојиновић З., Журкић А., Поповић В., Вуковић А., Недељковић С. - Пројектни задатак из предмета „Интелигентни технолошки системи“, Универзитет у Београду – Машински факултет, Београд, 2019

[27] Јокић А., Визуелно управљање интелигентног мобилног робота у функцији терминирања унутрашњег транспорта, мастер рад, Универзитет у Београду, Машински факултет, Београд 2017.

[28] Вег А., Шиниковић Г., Вег Е., Регодић М., Мали речник мехатронике, Универзитет у Београду, Машински факултет, Београд 2017

[29] Szeliski, R., Computer Vision: Algorithm and Application, Springer Science & Business Media, 2010.

[30] Meyer, Fernand, "Topographic distance and watershed lines," *Signal Processing* , Vol. 38, July 1994, pp. 113-125.,1994

[31] https://www.baslerweb.com/en/products/cameras/area-scancameras/ace/aca1920-25gm/ - датум последњег приступа 28.08.2019

[32] Лазаревић Д. Мишић М., Ђировић Б., Постојеће технике за сегментацију слике, 41. Национална конференција о квалитету, 2014.

[33] https://www.mathworks.com/help/deeplearning/ref/alexnet.html - последњи датум приступа 01.09.2019

[34] https://www.d2l.ai/chapter_computer-vision/semantic-segmentation-and-dataset.html - датум последњег приступа 01.09.2019

10. Прилог MATLAB програми

10.1 Главни програм за идентификацију технилошких ентитета на слици

```
close all, clc
clearvars -except CNN

load AlexNet_FINAL
CNN = AlexNet_FINAL;

inputSize = CNN.Layers(1).InputSize;

allImages = imageDatastore('Slike\Slike za testiranje
02.08','IncludeSubfolders', true,
'LabelSource','foldername');
i_num = 226; %401
I = imread(allImages.Files{i_num});
I = imresize(I, [360 640]);
% figure(), imshow(I)
Igray = rgb2gray(I);
% figure(), imshow(Igray)
Igray = double(Igray);

%% Gaus - blur
I_gaus = zeros(size(Igray));
filter = [1 2 1; 2 4 2; 1 2 1]*1.4;
for i = 2:size(Igray,1)-1
    for j = 2:size(Igray,2)-1
        I_gaus(i,j) = (Igray(i-1,j-1) * filter(1,1) +
Igray(i-1,j) * filter(1,2)  + Igray(i-1,j+1) *
filter(1,3)...
            + Igray(i,j-1) * filter(2,1) + Igray(i,j) *
filter(2,2)  + Igray(i,j+1) * filter(2,3) ...
            + Igray(i+1,j-1) * filter(3,1) + Igray(i+1,j) *
filter(3,2)  + Igray(i+1,j+1) * filter(3,3))/16;
    end
end

% I_gaus = imgaussfilt(Igray,2); % - drugi nacin

I_int = uint8(I_gaus);
% figure(),imshow(I_int)
```

```
%% Border control
I_gaus(1,1:size(I,2)) = I_gaus(2,1:size(I,2));
I_gaus(size(I,1),1:size(I,2)) = I_gaus(size(I,1)-
1,1:size(I,2));

I_gaus(1:size(I,1),1) = I_gaus(1:size(I,1),2);
I_gaus(1:size(I,1),size(I,2)) =
I_gaus(1:size(I,1),size(I,2)-1);
I_gaus = uint8(I_gaus);
% figure(),imshow(I_gaus)

%% Segmentation
a = zeros(size(I));
a = edge(I_gaus,'sobel','nothinning',0.012);
BW = a;
% figure(), imshow(BW)
BW = bwareaopen(BW,300);
% figure(),imshow(BW)

%% Edge connectivity
ED = 10;
for i = 1:size(BW,1)
    for j = 1:size(BW,2)
        if i <= ED
            if BW(i,j) == 1
                BW(1:i,j) = 1;
            end
        elseif i >= size(BW,1)-ED
            if BW(i,j) == 1
                BW(i:size(BW,1),j) = 1;
            end
        end

        if j <= ED
            if BW(i,j) == 1
                BW(i,1:j) = 1;
            end
        elseif j >= size(BW,2)-ED
            if BW(i,j) == 1
                BW(i,j:size(BW,2)) = 1;
            end
        end
    end
end
```

```
% figure(), imshow(BW)

%% Region clustering
BW = ~BW;
% figure(), imshow(BW)

bw = zeros(size(I));
bw = bwconncomp(BW,8);
bw_lab = zeros(size(I));
bw_lab = labelmatrix(bw);
RGB_label = zeros(size(I));
RGB_label = label2rgb(bw_lab, @copper, 'c', 'shuffle');
% figure(), imshow(RGB_label,'InitialMagnification','fit')

%% Extraction of labeled regions
n = bw.NumObjects;
group = zeros(size(bw_lab,1),size(bw_lab,2));
for x = 1:n
    for i = 1:size(bw_lab,1);
        for j = 1:size(bw_lab,2);
            if bw_lab(i,j) == x
                group(i,j) = 1;
            else
                group(i,j) = 0;
            end
        end
    end
    gr(:,:,x) = group;
end

BW = ~BW;
% figure(),imshow(BW)
hsv_obj = BW;
% figure(),imshow(BW)

%% RGB region analysis - za izdvajanje objekata od
okruzenja
RP = regionprops(bw_lab,'PixelList');

for i = 1:n
    pix_list = RP(i).PixelList'; % koordinate
    for j = 1:size(pix_list,2)
        rgb(1:3,j) = I(pix_list(2,j),pix_list(1,j),:);  % R
G B
```

```
    end

    Green(i) = (sum(rgb(2,:)))/(length(rgb(2,:))); % 2 - G

    if Green(i) < 120
       BW = BW + gr(:,:,i);
    end
    clear rgb
end

% figure(),imshow(BW)

%% HSV region analysis - za eliminisanje ivice okruzenja
HSV = rgb2hsv(I);
for i = 1:size(I,1)
    for j = 1:size(I,2)
        if hsv_obj(i,j) == 1
            hue = HSV(i,j,1);
            saturation = HSV(i,j,2);
            value = HSV(i,j,3);

            if saturation < 0.26
               BW(i,j) = 0;
            end

        end
    end
end

% figure(), imshow(BW);
BW = imfill(BW,'holes');
BW = bwareaopen(BW,300);
% figure(), imshow(BW);
BW_1 = BW;

%% Watershed
D = zeros(size(I));
D = -bwdist(~BW);
mask = imextendedmin(D,7);
% figure(); imshowpair(BW,mask,'blend')
D2 = zeros(size(I));
D2 = imimposemin(D,mask);
% figure(); imshow(D2,[]);
Ld2 = zeros(size(I));
```

```
Ld2 = watershed(D2);
% figure();imshow(label2rgb(Ld2))

BW(Ld2 == 0) = 0;
% figure(); imshow(BW)

%% Visualisation of the segmented image
bw_1 = bwconncomp(BW,8);
bw_lab_1 = labelmatrix(bw_1);
RGB_label = zeros(size(I));
RGB_label = label2rgb(bw_lab_1, @copper, 'c', 'shuffle');
figure(), imshow(RGB_label,'InitialMagnification','fit')


%% Wathershed check
BBB = B_BOX(I,BW);

for i = 1:size(BBB,1)
    obj = imcrop(I,BBB(i,:));
    obj_resize =
augmentedImageDatastore(inputSize(1:2),obj);
    [label(i), scores(i,:)] = classify(CNN, obj_resize);

end

[A,B,C]= unique(label,'stable'); % C ide redom 1 2 3 4 zbog
stable
n_1 = bw_1.NumObjects;
RPP = regionprops(bw_lab_1,'Area','Centroid');
j=1;
for i = 1:n_1
    if RPP(i).Area > 1500
        RP_1(j,:) = RPP(i).Centroid;
        j = j+1;
    end
end

for i = 1:size(C,1)
    if i+1 <= size(C,1)
        if C(i) == C(i+1)
            x_1 = round(RP_1(i,2));
            y_1 = round(RP_1(i,1));

            x_2 = round(RP_1(i+1,2));
```

```
            y_2 = round(RP_1(i+1,1));

            obj_1 = zeros(360,640);
            obj_2 = zeros(360,640);

            obj_1(x_1,y_1) = 1;
            obj_2(x_2,y_2) = 1;

            D1 = bwdist(obj_1, 'quasi-euclidean');
            D2 = bwdist(obj_2, 'quasi-euclidean');

            D_sum = D1 + D2;
            if min(min(D_sum)) < 250
                path_pixels = imextendedmin(D_sum,5);
            else
                path_pixels = zeros(size(I,1),size(I,2));
            end

            BW = BW + path_pixels;

        end
    end
end
BW = im2bw(BW);
% figure(); imshow(path_pixels)

figure(); imshow(BW)
BBB = B_BOX(I,BW);

figure('Resize','on'), imshow(I);
set(gcf, 'Position',  [460, 350, 640, 420])
hold on
if size(BBB,1) == 1
    title({['There is ', num2str(size(BBB,1)),' object in
the image'];''},'FontSize',12)
else
    title({['There are ', num2str(size(BBB,1)),' objects in
the image'];''},'FontSize',12)

end

clear label scores

%% Classify image
```

```
for i = 1:size(BBB,1)
    obj = imcrop(I,BBB(i,:));
    obj_resize =
augmentedImageDatastore(inputSize(1:2),obj);

    [label(i), scores(i,:)] = classify(CNN, obj_resize);
    LABEL = [char(label(i)),' -
',num2str(max(scores(i,:)*100),4),' %'];

    if mod(i,2) == 0
        text(BBB(i,1)+3,BBB(i,2)-7 +
BBB(i,4),char(LABEL),'Color','Black','FontSize',10,'FontWei
ght','bold')
    else
        text(BBB(i,1)+3,BBB(i,2)-7,char(LABEL),'Color',[0 0
0],'FontSize',10,'FontWeight','bold')
    end

    v(i) = rectangle('Position',BBB(i,:));
    set(v(i),'EdgeColor',[0.75 0 0]);
end

%% Object labeling
bw = zeros(size(I));
bw = bwconncomp(BW,8);
bw_lab = zeros(size(I));
bw_lab = labelmatrix(bw);
RGB_label = zeros(size(I));
RGB_label = label2rgb(bw_lab, @copper, 'c', 'shuffle');
% figure(), imshow(RGB_label,'InitialMagnification','fit')
```

10.2 Подпрограм за креирање оквира региона

```
function [BBB] = B_BOX(I,BW)

% Object labeling
bw = zeros(size(I));
bw = bwconncomp(BW,8);
bw_lab = zeros(size(I));
bw_lab = labelmatrix(bw); % tu se nalaze numerisani objekti
RGB_label = zeros(size(I));
RGB_label = label2rgb(bw_lab, @copper, 'c', 'shuffle');
% figure(), imshow(RGB_label,'InitialMagnification','fit')

% Extraction of labeled objects
n = bw.NumObjects;
group = zeros(size(bw_lab,1),size(bw_lab,2));
for x = 1:n
    for i = 1:size(bw_lab,1);
        for j = 1:size(bw_lab,2);
            if bw_lab(i,j) == x
                group(i,j) = 1;
            else
                group(i,j) = 0;
            end
        end
    end
    lab_obj(:,:,x) = group;
end

% Extraction of small and big objects
p = 1;
RP = regionprops(bw_lab(:,:,:),'Area');
for i = 1:size(lab_obj,3)
    if RP(i).Area > 1500 && RP(i).Area < 95000
        objects(:,:,p) = lab_obj(:,:,i);
        p = p+1;
    end
end

% Bounding Box

% figure, imshow(I);
% hold on;
position = zeros(size(objects,3),4);
```

```
for i = 1:size(objects,3)
    stats = regionprops(objects(:,:,i),'BoundingBox');
    box = stats.BoundingBox;

%         h(i) = rectangle('Position',box);
%         set(h(i),'EdgeColor',[.75 0 0]);

    position(i,1:4) = floor(box);
end

% title(['There are ', num2str(size(objects,3)),' objects
in the image'])

% Bigger Bounding Box
BBB = zeros(size(position,1),size(position,2));
N = 5;

BBB(:,1) = position(:,1) - N;
BBB(:,2) = position(:,2) - N;
BBB(:,3) = position(:,3) + N*2;
BBB(:,4) = position(:,4) + N*2;

for i = 1:size(BBB,1) % ogranicenje BBB - gore/levo
    if BBB(i,1) < 0
        BBB(i,1) = 1;
    end

    if BBB(i,2) < 0
        BBB(i,2) = 1;
    end
end

for i = 1:size(BBB,1) % ogranicenje BBB - dole/desno
    Width(i) = BBB(i,1) + BBB(i,3);
    Hight(i) = BBB(i,2) + BBB(i,4);

    if Width(i) > size(I,2)
        Width(i) = size(I,2); % j osa - x
        BBB(i,3) = Width(i) - BBB(i,1);
    end

    if  Hight(i) > size(I,1)
        Higth(i) = size(I,1); % i osa - y
        BBB(i,4) = Higth(i) - BBB(i,2);
```

```
    end
end
end
```

10.3 Програм за прикупљање обучавајућих парова

```
clear, close all, clc

load AlexNet_FINAL
CNN = AlexNet_FINAL;
inputSize = CNN.Layers(1).InputSize;

allImages = imageDatastore('Slike\Slike za testiranje
02.08','IncludeSubfolders', true,
'LabelSource','foldername');
% i_num = 23; size(allImages.Files,1)
for i_num = 1:size(allImages.Files,1)
I = imread(allImages.Files{i_num});
I = imresize(I, [360 640]);
% figure(), imshow(I)
Igray = rgb2gray(I);
Igray = double(Igray);

%% Gaus - blur
I_gaus = zeros(size(Igray));
filter = [1 2 1; 2 4 2; 1 2 1]*1.4;
for i = 2:size(Igray,1)-1
    for j = 2:size(Igray,2)-1
        I_gaus(i,j) = (Igray(i-1,j-1) * filter(1,1) +
Igray(i-1,j) * filter(1,2)  + Igray(i-1,j+1) *
filter(1,3)...
            + Igray(i,j-1) * filter(2,1) + Igray(i,j) *
filter(2,2)  + Igray(i,j+1) * filter(2,3) ...
            + Igray(i+1,j-1) * filter(3,1) + Igray(i+1,j) *
filter(3,2)  + Igray(i+1,j+1) * filter(3,3))/16;
    end
end

% I_gaus = imgaussfilt(Igray,2); % - drugi nacin

I_int = uint8(I_gaus);
% figure(),imshow(I_int)

%% Border control
I_gaus(1,1:size(I,2)) = I_gaus(2,1:size(I,2));
```

```
I_gaus(size(I,1),1:size(I,2)) = I_gaus(size(I,1)-
1,1:size(I,2));

I_gaus(1:size(I,1),1) = I_gaus(1:size(I,1),2);
I_gaus(1:size(I,1),size(I,2)) =
I_gaus(1:size(I,1),size(I,2)-1);
I_gaus = uint8(I_gaus);
% figure(),imshow(I_gaus)

%% Segmentation
a = zeros(size(I));
a = edge(I_gaus,'sobel','nothinning',0.012);
BW = a;
% figure(), imshow(BW)
BW = bwareaopen(BW,300);
% figure(),imshow(BW)

%% Edge connectivity
ED = 10;
for i = 1:size(BW,1)
    for j = 1:size(BW,2)
        if i <= ED
            if BW(i,j) == 1
                BW(1:i,j) = 1;
            end
        elseif i >= size(BW,1)-ED
            if BW(i,j) == 1
                BW(i:size(BW,1),j) = 1;
            end
        end

        if j <= ED
            if BW(i,j) == 1
                BW(i,1:j) = 1;
            end
        elseif j >= size(BW,2)-ED
            if BW(i,j) == 1
                BW(i,j:size(BW,2)) = 1;
            end
        end
    end
end
% figure(), imshow(BW)
```

```
%% Region clustering
BW = ~BW;
% figure(), imshow(BW)

bw = zeros(size(I));
bw = bwconncomp(BW,8);
bw_lab = zeros(size(I));
bw_lab = labelmatrix(bw);
RGB_label = zeros(size(I));
RGB_label = label2rgb(bw_lab, @copper, 'c', 'shuffle');
% figure(), imshow(RGB_label,'InitialMagnification','fit')

%% Extraction of labeled regions
n = bw.NumObjects;
group = zeros(size(bw_lab,1),size(bw_lab,2));
for x = 1:n
    for i = 1:size(bw_lab,1);
        for j = 1:size(bw_lab,2);
            if bw_lab(i,j) == x
                group(i,j) = 1;
            else
                group(i,j) = 0;
            end
        end
    end
    gr(:,:,x) = group;
end

BW = ~BW;
% figure(),imshow(BW)
hsv_obj = BW;
% figure(),imshow(BW)

%% RGB region analysis - za izdvajanje objekata od
okruzenja
RP = regionprops(bw_lab,'PixelList');

for i = 1:n
    pix_list = RP(i).PixelList'; % koordinate
    for j = 1:size(pix_list,2)
        rgb(1:3,j) = I(pix_list(2,j),pix_list(1,j),:);  % R
G B
    end
```

```
    Green(i) = (sum(rgb(2,:)))/(length(rgb(2,:))); % 2 - G

    if Green(i) < 120
       BW = BW + gr(:,:,i);
    end
    clear rgb
end

% figure(),imshow(BW)

%% HSV region analysis - za eliminisanje ivice okruzenja
HSV = rgb2hsv(I);
for i = 1:size(I,1)
    for j = 1:size(I,2)
        if hsv_obj(i,j) == 1
            hue = HSV(i,j,1);
            saturation = HSV(i,j,2);
            value = HSV(i,j,3);

            if saturation < 0.26
               BW(i,j) = 0;
            end

        end
    end
end

% figure(), imshow(BW);
BW = imfill(BW,'holes');
BW = bwareaopen(BW,300);
% figure(), imshow(BW);
BW_1 = BW;

%% Watershed
D = zeros(size(I));
D = -bwdist(~BW);
mask = imextendedmin(D,7);
% figure(); imshowpair(BW,mask,'blend')
D2 = zeros(size(I));
D2 = imimposemin(D,mask);
% figure(); imshow(D2,[]);
Ld2 = zeros(size(I));
Ld2 = watershed(D2);
% figure();imshow(label2rgb(Ld2))
```

```
BW(Ld2 == 0) = 0;
% figure(); imshow(BW)

%% Visualisation of the segmented image
bw_1 = bwconncomp(BW,8);
bw_lab_1 = labelmatrix(bw_1);
RGB_label = zeros(size(I));
RGB_label = label2rgb(bw_lab_1, @copper, 'c', 'shuffle');
figure(), imshow(RGB_label,'InitialMagnification','fit')


%% Wathershed check
BBB = B_BOX(I,BW);

for i = 1:size(BBB,1)
    obj = imcrop(I,BBB(i,:));
    obj_resize =
augmentedImageDatastore(inputSize(1:2),obj);
    [label(i), scores(i,:)] = classify(CNN, obj_resize);

end

[A,B,C]= unique(label,'stable'); % C ide redom 1 2 3 4 zbog
stable
n_1 = bw_1.NumObjects;
RPP = regionprops(bw_lab_1,'Area','Centroid');
j=1;
for i = 1:n_1
    if RPP(i).Area > 1500
        RP_1(j,:) = RPP(i).Centroid;
        j = j+1;
    end
end

for i = 1:size(C,1)
    if i+1 <= size(C,1)
        if C(i) == C(i+1)
            x_1 = round(RP_1(i,2));
            y_1 = round(RP_1(i,1));

            x_2 = round(RP_1(i+1,2));
            y_2 = round(RP_1(i+1,1));
```

```
            obj_1 = zeros(360,640);
            obj_2 = zeros(360,640);

            obj_1(x_1,y_1) = 1;
            obj_2(x_2,y_2) = 1;

            D1 = bwdist(obj_1, 'quasi-euclidean');
            D2 = bwdist(obj_2, 'quasi-euclidean');

            D_sum = D1 + D2;
            if min(min(D_sum)) < 250
                path_pixels = imextendedmin(D_sum,5);
            else
                path_pixels = zeros(size(I,1),size(I,2));
            end

            BW = BW + path_pixels;

        end
    end
end
BW = im2bw(BW);
% figure(); imshow(path_pixels)

figure(); imshow(BW)
BBB = B_BOX(I,BW);

figure('Resize','on'), imshow(I);
set(gcf, 'Position',  [460, 350, 640, 420])
hold on
if size(BBB,1) == 1
    title({['There is ', num2str(size(BBB,1)),' object in
the image'];''},'FontSize',12)
else
    title({['There are ', num2str(size(BBB,1)),' objects in
the image'];''},'FontSize',12)

end

clear label scores

%% Classify image
for i = 1:size(BBB,1)
    obj = imcrop(I,BBB(i,:));
```

```
    obj_resize =
augmentedImageDatastore(inputSize(1:2),obj);

    [label(i), scores(i,:)] = classify(CNN, obj_resize);
    LABEL = [char(label(i)),' -
',num2str(max(scores(i,:)*100),4),' %'];

    if mod(i,2) == 0
        text(BBB(i,1)+3,BBB(i,2)-7 +
BBB(i,4),char(LABEL),'Color','Black','FontSize',10,'FontWei
ght','bold')
    else
        text(BBB(i,1)+3,BBB(i,2)-7,char(LABEL),'Color',[0 0
0],'FontSize',10,'FontWeight','bold')
    end

    v(i) = rectangle('Position',BBB(i,:));
    set(v(i),'EdgeColor',[0.75 0 0]);
end

%% Object extraction
for i=1:size(BBB,1)
    obj = imcrop(I,BBB(i,:));
    obj_resize = imresize(obj, [227 227]);
    pause(0.1)

    if i == 1;
        Slika = ['T1_Levo1_Slika_',num2str(i_num),'.jpg'];
        folder = 'D:/Desktop/Machine 1';
        fullFileName = fullfile(folder, Slika);
        imwrite(obj_resize, fullFileName);
    elseif i == 2;
        Slika = ['T1_Levo2_Slika_',num2str(i_num),'.jpg'];
        folder = 'D:/Desktop/Machine 2';
        fullFileName = fullfile(folder, Slika);
        imwrite(obj_resize, fullFileName);
    elseif i == 3;
        Slika = ['T1_Levo3_Slika_',num2str(i_num),'.jpg'];
        folder = 'D:/Desktop/Machine 3';
        fullFileName = fullfile(folder, Slika);
        imwrite(obj_resize, fullFileName);
    elseif i == 4;
        Slika = ['T1_Levo4_Slika_',num2str(i_num),'.jpg'];
        folder = 'D:/Desktop/Machine 4';
```

```
        fullFileName = fullfile(folder, Slika);
        imwrite(obj_resize, fullFileName);
    else i == 5;
        Slika = ['T1_Levo5_Slika_',num2str(i_num),'.jpg'];
        folder = 'D:/Desktop/Black Marker';
        fullFileName = fullfile(folder, Slika);
        imwrite(obj_resize, fullFileName);
    end
end
clearvars -except allImages CNN
end
```

10.4 Програм за „дообучавање“ готових модела CNN-а

```
clear, close, clc
%% Load pretrained CNN
net = alexnet;
layers = net.Layers;
% sz = net.Layers(1).InputSize;
%% Modify the network
layers(23) = fullyConnectedLayer(4,'Name','fc8');
layers(25) = classificationLayer('Name','output');

%% Load image and set up data
trainingImages= imageDatastore('D:\Desktop\Google
Drive\Sava_Nedeljkovic_1819\Testovi_2.Jul\Features','Includ
eSubfolders', true, 'LabelSource','foldername');
testImages = imageDatastore('D:\Desktop\Google
Drive\Sava_Nedeljkovic_1819\Slike za testiranje
02.08\Features 02.08','IncludeSubfolders', true,
'LabelSource','foldername');

%% Re-train the Network
opts =
trainingOptions('sgdm','InitialLearnRate',0.001,'LearnRateD
ropFactor', 0.3, 'MaxEpochs',5, 'MiniBatchSize', 100,...
       'Plots','training-progress');
Alex_net = trainNetwork(trainingImages, layers, opts);

%% Measure network accuracy
predictedLabels = classify(Alex_net, testImages);
accuracy = mean(predictedLabels == testImages.Labels)
%% Confusion matrix
```

```
[CM,ORDER] =
confusionmat(testImages.Labels,predictedLabels);
confusionchart(CM,ORDER);
title('Matrica Odlucivanja');

save Alex_net
AlexNet_FINAL = Alex_net;
save AlexNet_FINAL AlexNet_FINAL
%save ('soljevi','layers')
```

10.5 Програм за компарацију постојећих метода сегментације

```
clc, clear all, close all

I = imread('Test.jpg');
Igray = rgb2gray(I);

figure(1); imshow(I)
figure(2); imhist(Igray(:,:,:))

a = edge(Igray,'roberts',0.017);
b = edge(Igray,'sobel',0.017);
c = edge(Igray,'prewitt',0.017);
d = edge(Igray,'canny',0.017);
e = edge(Igray,'log',0.017);

figure(3)
subplot(2,3,1), imshow(I), title('Original')
subplot(2,3,2), imshow(a), title('Roberts')
subplot(2,3,3), imshow(b), title('Sobel')
subplot(2,3,4), imshow(c), title('Prewitt')
subplot(2,3,5), imshow(d), title('Canny')
subplot(2,3,6), imshow(e), title('Log')
```

10.6 Програм за аквизицију слика са леве и десне камере

```
close all, clc

load CNN_all

%% Setting up Video
% Left camera
vid = videoinput('gentl',1,'BayerGB8');%, 'RGB24_640x480')
BILO :YUY2_320x240
src = getselectedsource(vid);
triggerconfig(vid,'manual');

set(vid,'FramesPerTrigger',15);

set(vid,'TriggerRepeat', Inf);

set(vid,'ReturnedColorSpace','rgb')

set(vid, 'ReturnedColorSpace','RGB')

src.BalanceWhiteAuto = 'Once';

start(vid)
%%%%%% Za gledanje snimka
vidRes = vid.VideoResolution;
nBands = vid.NumberOfBands;
hImage = image( zeros(vidRes(2), vidRes(1), nBands) );
preview(vid, hImage);

% Right camera
vid1 = videoinput('gentl',2,'BayerGB8');
src1 = getselectedsource(vid1);
triggerconfig(vid1,'manual');

set(vid1,'FramesPerTrigger',15);

set(vid1,'TriggerRepeat', Inf);

set(vid1,'ReturnedColorSpace','rgb')

set(vid, 'ReturnedColorSpace','RGB')

src1.BalanceWhiteAuto = 'Once';
```

```
start(vid1)
%%%%%% Za gledanje snimka
vidRes1 = vid1.VideoResolution;
nBands1 = vid1.NumberOfBands;
figure(2)
hImage1 = image( zeros(vidRes1(2), vidRes1(1), nBands1) );
preview(vid1, hImage1);

%% Take Image
slika = getsnapshot(vid);
slika1 = getsnapshot(vid1);

I1 = slika(:,:,:);
I2 = slika1(:,:,:);

I1 = imresize(I1,[400 400]);
I2 = imresize(I2,[400 400]);

[BW1,BBB1] = segmentation(I1);
[BW2,BBB2] = segmentation(I2);


figure, imshow(I1);
hold on
%title(['There are ', num2str(size(obj_position,1)),'
objects in the image'])

for i = 1:size(BBB1,1)
    v(i) = rectangle('Position',BBB1(i,:));
    set(v(i),'EdgeColor',[.75 0 0]);

end

for i=1:size(BBB1,1)
    obj = imcrop(I,BBB1(i,:));
    obj_resize = imresize(obj, [227 227]);

    label(i) = classify(CNN_all, obj_resize);

    text(BBB1(i,1)-12,BBB1(i,2)-
12,char(label(i)),'Color','Black')

end
```