Зборник радова Факултета техничких наука, Нови Сад

UDK: 004.42
DOI: https://doi.org/10.24867/08BE23Ilicic

ПРЕПОЗНАВАЊЕ ГРАФИЧКИХ СИМБОЛА НА ШЕМАТСКИМ ДИЈАГРАМИМА

IDENTIFICATION OF GRAPHIC SYMBOLS ON SCHEMATIC DIAGRAMS

Вукашин Иличић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *Овим радом обухваћено је истраживање различитих приступа и алгоритама који су везани за препознавање симбола на шематским дијаграмима. Описан је поступак на којем се заснива имплементација апликације која омогућава кориснику да пронађи жељене симболе на изабраној шеми. Приликом имплементације коначног решења коришћен је C# програмски језик. Током развоја апликације урађене су детаљне анализе и тестирања која испитују њене перформансе.*

Кључне речи: *Вештачка интелигенција, машинско учење, детекција објеката, шематски дијаграм*

Abstract – *This paper covers the study of different approaches and algorithms related to the identification of symbols in schematic diagrams. The paper describes in detail the procedure on which the implementation of an application is based, which allows the user to detect desired symbols on an input scheme. The C# programming language was used in the final solution. During the development of the program, detailed analyzes and tests were conducted to test the application performances.*

Keywords: *Artificial Intelligence, Machine learning, Object-detection, Schematic diagram*

1. УВОД

Анализа слика као област је позната од раније и тренутно је врло популарна пошто је нашла широку примену у индустрији и различитим пословима који су постали саставни део наших живота. У овој области постоје различити приступи и технике које се примењују. Препознавање симбола на сликама може да се реализује на различите начине помоћу технике машинског учења или помоћу принципа директне претраге. За реализацију решења коришћена су оба приступа, где је установљено да решење које се заснива на употреби неуронских мрежа не даје очекиване резултате. Главни разлог за недовољно добре резултате је ограничена количина слика за обуку неуронске мреже. Код реализације решења помоћу неуронских мрежа коришћена је ***Google***-ова платформа ***TensorFlow***, намењена за машинско учење. Она пружа разноврсне могућности код обучавања неуронских мрежа. Након што је установљено да због недовољне количине података резултати остварени помоћу машинског учења нису довољно добри у обзир је узето решење које се заснива на принципу „клизача“ који пролази кроз улазну шему и тражи елементе одређеног типа. Крајње решење реализовано је помоћу принципа директне претраге која користи структуру пирамиде слика. Структура пирамиде слика омогућава брже извршавање процеса тражења одређеног типа јер користи посебну структуру која смањује опсег претраге. Имплементирана апликација је осмишљена да омогући кориснику да учита жељену слику на којој је представљена шема са различитим елементима које је потребно препознати. После одабира дела слике на којој се налази жељени тип елемента извршава се претрага означеног типа. Као резултат претраге добија се учитана слика са претраженим елементима који припадају истом типу као и означени елемент којег је одабрао корисник.

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био доц. др Милан Гаврић.

2. ОПИС АНАЛИЗИРАНИХ ТЕХНОЛОГИЈА

Поред краћег описа имплицитно или експлицитно су наведене неке од предности и мана које долазе са њиховим коришћењем. Описане су само технологије и приступи који су највише утицали на имплементацију решења.

2.1. COMPUTER VISION

Научници и истраживачи који су се бавили темом ***computer vision***-а, развили су различите технике засноване на математици за враћање тродимензионалног облика и изгледа објеката на сликама [1]. Захваљујући њима, данас постоје поуздане технике које омогућавају да се на основу одређеног броја фотографија које су дате у две димензије, створи 3D модел који верно репрезентује објекат са фотографија. Поред наведене функционалности могуће је пратити и кретање особе на видео снимку па чак пребројати и именовати људе са фотографије помоћу технике препознавања лица.

2.2 *AI*, МАШИНСКО УЧЕЊЕ И *DEEP LEARNING*

Вештачка интелигенција је као област у рачунарству настала педесетих година прошлог века. Вештачка интелигенција се може посматрати и као напор да се аутоматизују интелектуални задаци које људи могу уобичајено да обављају. Као таква, област ***AI*** (***artificial intelligence***) обухвата машинско учење и ***deep learning*** [3]. Машинско учење представља програмирање рачунара тако да могу да уче из података. Сличну дефиницију о машинском учењу написао је и Arthur Samuel (1959), „Машинско учење је област која рачунарима даје могућност учења без експлицитног програмирања“ [4].

Deep learning представља специфично поље машинског учења, где се усвајање новог знања

заснива на принципу неуронских мрежа које се састоје из више слојева. Модерне методе ***deep learning***-а укључују десетине или чак стотине узастопних слојева који се обучавају помоћу одабраних података за тренинг. Да би се слике могле обрађивати помоћу рачунара, могуће је применити концепте машинског учења и неуронских мрежа. Процес препознавања објеката на сликама помоћу машинског учења је сложен и захтева доста времена и примера за учење.

2.3. ПИРАМИДА СЛИКА

За имплементацију апликације која може да препознаје одабране симболе, коришћена је посебна структура у којој се улазна слика за претрагу скалира на мање слике. На тај начин се добија структура у облику пирамиде помоћу које је могуће ефикасније урадити претрагу жељених симбола. Овај принцип представља оптимизацију за такозвану директну претрагу шаблона на сликама.

Пирамида се састоји од одређеног броја нивоа. Први односно основни ниво пирамиде представља оригиналну слику, док сваки следећи ниво настаје умањивањем слике са претходног нивоа [2]. Процес корелације, односно поређења шаблона који се претражује и саме слике на којој треба пронаћи шаблон започиње од врха пирамиде. Због тога што највиши ниво садржи најмању слику, омогућено је брже извршавање претраге. Кроз процес поређења утврђују се такозване хипотезе о томе где би тражени шаблон могао да се налази.

Хипотезе које су добијене на вишим слојевима, проверавају се на нижим. Овакав приступ омогућава да се мали део око грубе процене са вишег нивоа испитује у оквиру нижих нивоа. Овај алгоритам пружа смањени опсег претраге и помоћу њега могу брзо и лако да се разврстају делови слике што доводи до оптимизације која резултује краћим временом извршавања процеса претраге.

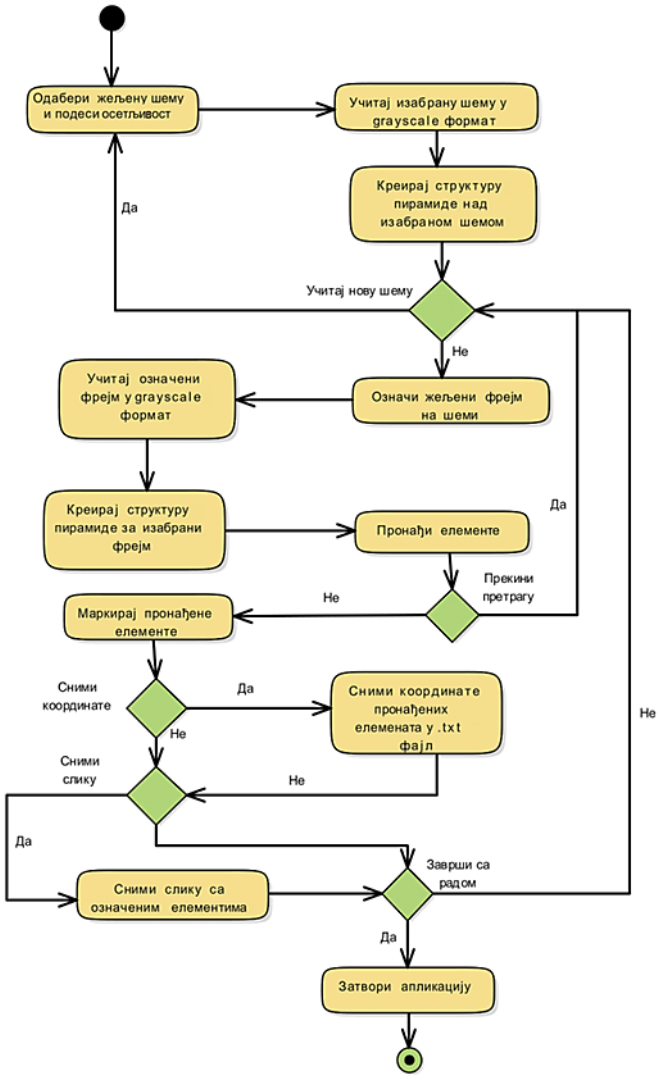
3. ДИЗАЈН РЕШЕЊА

У оквиру овог поглавља укратко је представљена реализација решења за задати проблем. Треба напоменути да се током имплементације решења мењао приступ решавања због одређених ограничења која су наведена у наставку. Улаз у програм представљају слике, које се конвертују у ***grayscale*** формат. Апликација је направљена тако да ради са ***grayscale*** сликама (Слика 1), међутим корисник може да бира који приказ слике жели приликом рада у програму (***Grayscale*** или RGB). Интерно у меморији учитана слика је представљена помоћу матрице, чија поља имају вредности у опсегу од 0 до 255.

Свака вредност у матрици представља ***grayscale*** вредност одређеног пиксела на изабраној шеми. Када се слика за претрагу учита, кориснику је омогућено да одабере елемент који жели да се претражи. Означени фрејм се такође претвара у ***grayscale*** матрицу вредности интерно у меморији.

После учитавања жељене слике и ***frame***-а следећи корак представља креирање пирамиде. Пирамида се посебно креира за учитану слику и за селектовану област која се претражује. Креирање пирамиде се ради тако што се вредности четири суседна пиксела сабирају и деле са четири. На тај начин се добија њихова средња вредност.

Са овим кораком добијају се слојеви који су по висини и по ширини за дупло мањи од свог родитељског слоја од којег су настали. На тај начин новонастали слој је по броју пиксела четири пута мањи у односу на



Слика 1. *Алгоритам рада апликације*

родитељски слој. Број слојева у структури пирамиде се подешава ручно у зависности од резолуције улазне слике. Оваква структура је погодна због перформанси претраге која представља следећи корак у целокупном процесу. Процес претраге започиње од врха пирамиде учитане слике, односно од најмањег слоја. У оквиру најмањег слоја у којем се ради претрага, користи се фрејм са врха пирамиде фрејмова.

Фрејм који одговара врху пирамиде учитане шеме пролази кроз комплетну шему на врху структуре са кораком померања од једног пиксела. Процес поређења фрејма са одговарајућим делом шеме се ради тако што се пореде ***grayscale*** вредности боја код пиксела. Сваки пиксел на фрејму има одговарајући

пиксел на посматраном делу шеме са којим се пореди. На основу разлике вредности пиксела са шеме и фрејма, прорачунава се укупна разлика помоћу које се одлучује да ли је на шеми пронађен задати тип елемента. Помоћу параметра осетљивости поставља се горња граница која одређује колико процентуално посматрани део шеме може да се разликује од задатог елемента. Уколико је проценат различитости поређења мањи од вредности дефинисаног параметра осетљивости, онда се анализирани део шеме сматра припадником траженог типа. Када се претрага горњег слоја уради добијају се области, односно координате потенцијалних елемената који одговарају изабраном фрејму. У нижим слојевима уместо да се претрага ради од почетка до краја, она се врши само у областима који су добијени у вишим слојевима.

На тај начин се смањује област претраге на области које су од значаја за тражени резултат. Последњи корак представља постављање маркера који уоквирују пронађене елементе на слици. Резултат претходног корака представља листу координата пронађених елемената. На сваку од пронађених координата поставља се по један маркер (правоугаоник). Као резултат добија се слика са уоквиреним елементима који су претраживани. Поред слике са означеним елементима у склопу апликације корисник има опцију да експортује добијене координате у излазни текстуални фајл. У наставку ће бити објашњена имплементација компоненти које су од значаја за рад апликације.

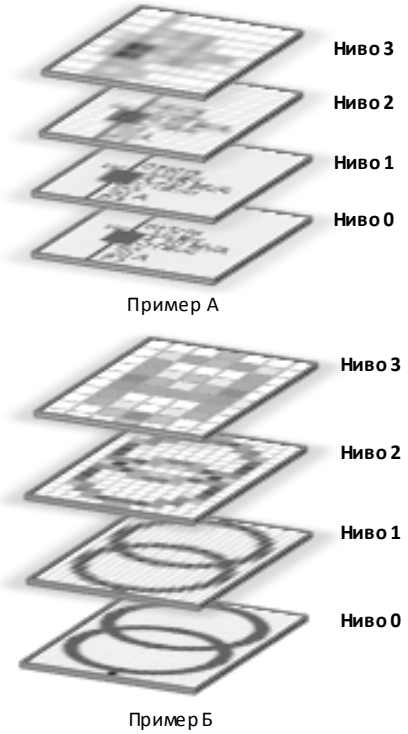
4. ИМПЛЕМЕНТАЦИЈА

Апликација је имплементирана по MVVM (model-view-viewmodel) патерну. Што се тиче view компоненти, апликација се састоји од основног *MainView*-а на којем се налазе сви потребни *widget*-и преко којих корисник остварује жељене радње. У склопу Main-ViewModel-а користе се различите класе дефинисане у моделу које се користе ради постизања жељених функционалности. Једна од главних класа из модела представља DataWorker класа која садржи податке о учитаној слици и означеном фрејму који су представљени у облику матрице са вредностима од 0 до 255. Претходно споменуте две матрице презентују ***grayscale*** формат слика са којима се ради у програму. Кориснику је омогућено да учитава растерске слике које могу бити у .jpg или .png формату.

Што се тиче метода у DataWorker класи, она садржи основне методе за учитавање података за изабрану слику у интерну структуру која презентује ***grayscale*** формат те слике. Након што се дефинишу сви потребни подаци везани за означени фрејм и слику на којој се врши претрага, могуће је започети процес претраге. Процес претраге се започиње тако што се прво креира структура пирамиде помоћу класе PyramidImageHelper. Споменута класа садржи две листе у којима су смештене матрице које презентују слику шеме и слику фрејма у различитим нивоима. Пирамида слика се креира тако што се на основу оригиналне слике шеме и слике фрејма креирају наредни скалирани нивои структуре пирамиде. Након креирања структуре пирамиде, започиње се процес тражења елемената који су означени помоћу изабраног фрејма. Процес претраживања креће од врха пирамиде где се добијени резултати користе у нижем слоју ради постизања истог циља код слике са већом резолуцијом. Процес претраге се понавља док се не заврши претрага на слици са највишом резолуцијом која се налази на најнижем слоју пирамиде. Сваком слоју пирамиде са скалираном сликом која се претражује одговара један слој пирамиде у којој су скалиране слике изабраног фрејма.

На овај начин се омогућава да се одбаце делови за претрагу који нису од значаја, односно да буду претражени само делови на слици који потенцијално садрже тражене елементе. Помоћу оваквог приступа претраге омогућава се мањи број итерација, што доводи до тога да целокупан процес претраге буде временски краћи. Слика 2 приказује пикселизацију елемената по слојевима. Пример А са слике приказује пикселизацију дела шеме, док је у примеру Б приказана пикселизација означеног елемента.

У оба примера структура пирамиде садржи четири нивоа. Са слике (Слика 2) се може уочити губитак својстава код приказаних елемената са порастом броја нивоа.



Слика 2. *Шематски приказ пикселизације (А) Пикселизације дела шеме, (Б) Пикселизација означеног елемента*

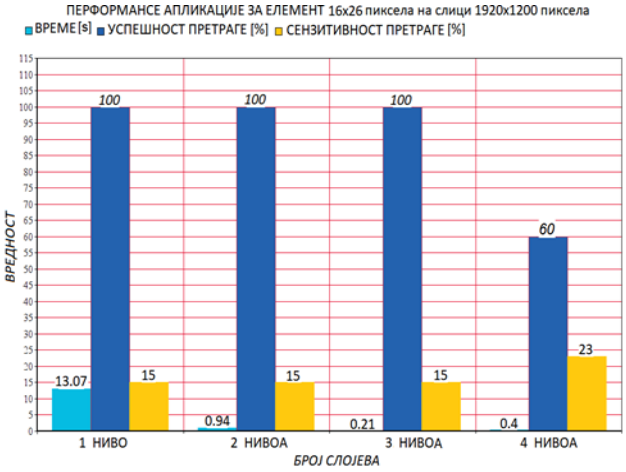
Губитак својстава код елемената утиче на успешност претраге, што је приказано и на графику перформанси (Слика 3). Са графика се може видети да код претраге елемената помоћу структуре пирамиде са четири нивоа успешност је смањена на 60%. Употреба структуре пирамиде смањује време извршавања али пикселизација која се јавља услед скалирања може да утиче на успешност претраживања. Због тога је битно проценити број нивоа за претрагу на основу величине и

својстава претраживаних елемената на изабраној шеми.

5. МЕРЕЊЕ ПЕРФОРМАНСИ РЕШЕЊА

У овом поглављу размотрене су перформансе рада апликације. Брзина извршавања претраге зависи од више фактора (Слика 3). Основни фактор који утиче на брзину претраживања јесте величина улазне слике. Апликација је тестирана за слике које су у ***full HD*** резолуцији. У зависности од резолуције потребно је подесити одређени број нивоа који ће се користити за претрагу. Код резолуције 1920 x 1080 пиксела, тестирање је показало да се најбоље перформансе могу постићи када се користе пирамиде са два или три нивоа. Осим резолуције улазне слике на брзину претраге утиче и величина претраживаног елемента. Уколико је резолуција претраживаног елемента већа онда је и време претраживања дуже.

Осим времена претраге, на графику су приказане и информације о успешности претраге као и осетљивости. Поред споменутих фактора и параметара на претрагу утиче и осетљивост коју подешава корисник. Подешавање осетљивости претраге је корисно, јер поспешује саму претрагу и на тај начин се остварују бољи резултати рада. На пример претраживани елементи истог типа имају своју специфичност која их разликује од других елемената тог типа. Због међусобне разноликости између елемената истог типа, кориснику је обезбеђено да сам може да подеси колика ће осетљивост да се користи током претраге. У зависности од геометрије и разноликости претраживаних елемената биће употребљена већа или мања осетљивост. Да би се искористили номинални параметри за претрагу, врло често је потребно урадити претрагу неколико пута да би могло да се закључи које вредности параметара су најбоље за разматрани случај претраге**.** Перформансе имплементиране апликације су тестиране на рачунару који има Intel-ов i3 процесор (3.2 GHZ) са два језгра као и 32GB RAM меморије.



Слика 3. *Измерене перформансе*

6. ЗАКЉУЧАК

Стручњаци који се баве машинским учењем и неуронским мрежама сматрају да ће вештачка интелигенција бити све присутнија у животу човека. Да би се ова визија остварила на прави начин врло је важно разумети начин на који поједине ствари функционишу, јер у супротном последице могу бити трајне, поготово када је реч о ***AI***. Овај рад обухвата само део целокупне области и као такав може да послужи као основа и инспирација за даља проучавања. Одређене специфичности које карактеришу решавани проблем, довеле су до тога да се уз употребу ***C#*** програмског језика имплементира алгоритам који ће омогућити потребну функционалност без да се користи машинско учење и ***deep learning***. Главни разлог због којег се користи другачији приступ је мали број слика које могу да се употребе за обуку неуронске мреже. Као прихватљиво решење за овај проблем показала се и оптимизована директна претрага која користи структуру пирамиде слика која је објашњена у претходном поглављу.

Решење које је имплементирано по споменутом принципу садржи и одређене недостатке. Приликом претраге користе се слике у ***grayscale*** формату, што уводи одређено ограничење. Боје које се јасно разликују у ***RGB*** формату, врло су сличне приликом конверзије у одговарајућу нијансу сиве боје. Овај недостатак долази до изражаја када су елементи истог типа представљени различитим бојама у ***RGB*** формату, које приликом конверзије у ***grayscale*** формат дају сличне нијансе сиве боје. Овај недостатак може да се реши помоћу различитих апликација као што је ***Photoshop*** који може помоћу ***color-range*** опције да одбаци делове слике обојене у одређену боју. Тиме би се омогућило да се претрага жељеног елемента ради независно од боје у RGB формату. Тема није у целости затворена и може да послужи као платформа за нова истраживања и радове.

7. ЛИТЕРАТУРА

[1] Szeliski Richard, “Computer Vision”, Springer, 2010.

[2] Marco Alexander Treiber, “An Introduction to Object Recognition”, 2010.

[3] F. Chollet, "Deep Learning with Python", Manning Publications, 2017.

[4] A. Géron, "TensorFlow" 2nd Edition, O'Reilly Media, 2019.

Кратка биографија:

Вукашин Иличић рођен је у Новом Саду 1995. године. Завршио је основну школу „Светозар Марковић Тоза“ 2010. године. Завршио је гимназију „Светозар Марковић“, општи смер, у Новом Саду 2014. године. Исте године је уписао основне академске студије на Факултету техничких наука, смер примењено софтверско инжењерство, у Новом Саду. На основним академским студијама дипломирао је 2018. године са просечном оценом 9,13. Мастер студије је уписао 2018. године и испунио је све обавезе и положио све испите предвиђене студијским програмом са просечном оценом 9,83.