



**УНИВЕРЗИТЕТ У НИШУ  
ЕЛЕКТРОНСКИ ФАКУЛТЕТ  
КАТЕДРА ЗА РАЧУНАРСТВО**



**Мастер рад**

**Реализација апликације за сепарацију елемената  
текста из скенираних докумената и генерисање  
базе карактера различитих формата**

**Марко Ђокић**

Ниш, 2024.

**УНИВЕРЗИТЕТ У НИШУ**  
**ЕЛЕКТРОНСКИ ФАКУЛТЕТ**  
**КАТЕДРА ЗА РАЧУНАРСТВО**

**Мастер рад**

**Реализација апликације за сепарацију елемената  
текста из скенираних докумената и генерисање  
базе карактера различитих формата**

Задатак: Истражити проблематику и методе које се користе за обраду слика у циљу побољшања тачности препознавања текста из скенираних докумената (*Optical Character Recognition - OCR*). Теоријски објаснити примену филтера на сликама, као и принцип рада алгоритама за препознавање знакова. Прегледати доступна решења, њихове предности и недостатке, као и резултате различитих *OCR* система у пракси. У практичном делу развити апликацију која омогућава обраду слика и препознавање текста, са могућношћу чувања привремених резултата препознатих карактера у базу карактера. Имплементирати проучене технике кроз реализацију апликације.

**Ментор:**

Проф. др. Владан Вучковић

**Кандидат:**

Марко Ђокић, 1022

**Комисија:**

1. \_\_\_\_\_

Датум пријаве: \_\_\_\_\_

2. \_\_\_\_\_

Датум предаје: \_\_\_\_\_

3. \_\_\_\_\_

Датум одбране: \_\_\_\_\_

# Реализација апликације за сепарацију елемената текста из скенираних докумената и генерисање базе карактера различитих формата

## Сажетак

Рад се бави процесом обраде слика са посебним акцентом на препознавање текста са скенираних докумената. Уводна целина је посвећена дигиталној обради слике, а главни део рада посвећен је препознавању текста са слика, познатијем као оптичко препознавање карактера (*Optical Character Recognition* - *OCR*). Након кратког увода у историју *OCR* технологије, обрађени су кључни кораци за успешно препознавање карактера у тексту применом модерних техника. Детаљно су описани савремени *OCR* системи, њихове предности и ограничења. У раду је приказана и компарација између постојећих решења и решења имплементираног у овом истраживању.

Практичан део овог рада је развој апликације за генерисање базе карактера, која користи *OCR* технологију за препознавање карактера са слика и креирање базе карактера. Ова апликација омогућава корисницима да сниме или прочитају слике са текстом, побољшају их, препознају карактере помоћу *OCR* система и аутоматски генеришу дигиталне фонтове односно базе карактера. Таква апликација може имати широку примену у дизајну, типографији, архивирању докумената и другим областима где је персонализација фонтова од значаја, такође резултати ове апликације могу бити готови узорци за неке алгоритме за даљу обраду и машинско учење. Апликација је развијена коришћењем напредних техника обраде слике и машинског учења, како би се обезбедила висока тачност препознавања карактера и генерисања фонтова.

**Кључне речи:** обрада слике, *Tesseract*, *OCR*, *OpenCV*, компресија слике

# *Implementation of application for text elements separation from scanned documents and generation of a character database of various formats*

## **Abstract**

*This work focuses on the process of image processing with a special emphasis on text recognition from scanned documents. An introductory section is dedicated to digital image processing, while the main part of the paper focuses on text recognition from images, commonly known as Optical Character Recognition (OCR). After a brief introduction to the history of OCR technology, the key steps for successful character recognition in text using modern techniques are addressed. Contemporary OCR systems, their advantages, and limitations are described in detail. The paper also provides a comparison between existing solutions and the solution implemented in this research.*

*The practical part of this work involves the development of an application for generating a character database, which uses OCR technology to recognize characters from images and create a character database. This application allows users to capture or load images with text, to enhance image, recognize characters using the OCR system, and automatically generate digital fonts or character databases. Such an application can have broad applications in design, typography, document archiving, and other areas where font personalization is important. Additionally, the results of this application can serve as ready-made samples for certain algorithms used in further processing and machine learning. The application is developed using advanced image processing and machine learning techniques to ensure high accuracy in character recognition and font generation.*

**Keywords:** *image processing, Tesseract, OCR, OpenCV, image compression*

## Садржај

<b>1. Увод.....</b>	<b>7</b>
<b>2. Дигитална обрада слике.....</b>	<b>8</b>
2.1. Узорковање и квантизација.....	8
2.2. Хистограм слике.....	9
2.3. <i>Grayscale</i> филтер.....	11
2.4. Интерполација слике.....	12
<b>3. Оптичко препознавање карактера.....</b>	<b>13</b>
3.1. <i>OCR</i> са машинским учењем.....	14
3.2. <i>OCR</i> са дубоким учењем.....	14
3.3. <i>OCR</i> алгоритам.....	15
3.4. Како <i>OCR</i> ради.....	16
3.4.1. Препознавање узорака.....	16
3.4.2. Детекција карактеристика.....	17
3.5. Како се ради са <i>OCR</i> алатом у пракси.....	18
<b>4. Технологије и библиотеке.....</b>	<b>19</b>
4.1. <i>Tesseract</i> .....	19
4.1.1. Уклапање основне линије.....	19
4.1.2. Детекција фиксне ширине слова и одсецање.....	20
4.1.3. Променљива ширина у тексту.....	20
4.1.4. Препознавање речи.....	20
4.1.5. Одсецање повезаних карактера.....	21
4.1.6. Повезивање изломљених карактера.....	21
4.1.7. Слични <i>OCR</i> алати.....	21
4.2. <i>Emgu.CV</i> .....	24
4.3. <i>Windows Forms</i> .....	25
4.4. Верзије и покретање.....	26
<b>5. Дијаграми апликације.....</b>	<b>27</b>
5.1. Дијаграм тока комплетног процеса обраде.....	27
5.2. Дијаграм случајева коришћења.....	28
5.3. Секвенцијални дијаграми.....	29
5.4. Дијаграм <i>Windows Form</i> компоненти.....	31
5.5. Дијаграм активности.....	32
5.6. Дијаграм протока података.....	33
<b>6. Имплементација.....</b>	<b>34</b>
6.1. Учитавање докумената.....	34
6.2. Бинаризација листе слика.....	34
6.3. Имплементација екстракције карактера.....	35
6.4. Компресовање и чување карактера у базу.....	36
<b>7. Илустрација имплементације.....</b>	<b>38</b>

7.1. Учитавање скенираног документа.....	38
7.2. Филтери и бинаризација.....	39
7.3. Сепарација карактера.....	41
7.4. Формирање базе карактера.....	44
7.5. Уређивање базе карактера.....	46
7.6. Нормализација.....	50
7.7. Приказ формата у <i>ASCII</i> редоследу.....	52
7.8. Додатни део апликације.....	53
<b>8. Закључак.....</b>	<b>55</b>
<b>9. Литература.....</b>	<b>56</b>
<b>10. Списак слика и листинга.....</b>	<b>57</b>

# 1. Увод

Технике дигиталне обраде слике развијене су 1960-их година у *Bell* лабораторијама, *Jet Propulsion* лабораторији, Масачусетс институту за технологију, универзитету у Мериленду и још неколико других истраживачких установа. То је примењивано на сателитским сликама, за медицинска снимања, видеотелефонију, препознавање карактера и унапређење фотографија. Сврха ране обраде слика била је побољшање квалитета слике. Циљ је био побољшање визуелног ефекта за људе. У обради слике, улаз је слика ниског квалитета, а излаз је слика са побољшаним квалитетом. Уобичајене методе обраде слике укључују побољшање слике, рестаурацију, кодирање и компресију. Прва успешна примена била је у америчкој лабораторији *Jet Propulsion*. Користили су технике обраде слике као што су геометријска корекција, градациона трансформација и уклањање шума. Било је потребно обрадити хиљаде фотографија које је снимио свемирски детектор Ренџер (*Ranger*) 7 1964. године, слике положаја Сунца и Месеца. Утицај успешног мапирања површине Месеца помоћу рачунара био је огроман. Касније је на скоро 100.000 фотографија које је послала летелица извршена сложенија обрада слике, тако да су добијене топографске карте, карте у боји и панорамске слике Месеца.

Трошкови обраде су били прилично високи с обзиром на тадашњу рачунарску опрему. То се дешавало 1970-их година, када је дигитална обрада слике полако постајала распрострањенија, него раније, захваљујући јефтинијим рачунарима и наменској опреми. Дошло је до тога да се слике обрађују у реалном времену за неке специфичне проблеме, као што су на пример конверзије телевизијских стандарда. Са брзим рачунарима и процесорима сигнала доступним у 2000-им годинама, дигитална обрада слике постала је најчешћи облик обраде слика и углавном се користи јер није само најприлагодљивији метод, већ и најјефтинији.

Прва камера је направљена 1975. године, али није снимила прву дигиталну фотографију. Прва дигитална фотографија заправо је настала 20 година раније, 1957. године, када је *Russell Kirsch* направио дигиталну слику резолуције 176x176 пиксела скенирањем фотографије свог тромесечног сина. Ниска резолуција била је резултат чињенице да рачунар који су користили није био способан да складишти више информација.

## 2. Дигитална обрада слике

### 2.1. Узорковање и квантизација

Сензори производе слику у облику аналогног сигнала, у обради дигиталних слика, сигнали снимљени из физичког света морају бити преведени у дигитални облик. Да би слика била погодна за дигиталну обраду, функција слике  $f(x,y)$  мора бити дигитализована и просторно и у амплитуди. Овај процес дигитализације обухвата два главна поступка а то су узорковање (*Sampling*) односно дигитализација координатних вредности и квантизација (*Quantization*) односно дигитализација амплитудних вредности.

Дигитална слика је дводимензионална функција  $f(x,y)$ , где  $x$  и  $y$  означавају положај на слици. Функција  $f(x,y)$  садржи дискретну вредност која се назива интензитет. Дигитална слика садржи скуп елемената који се називају пиксели или сликовни елементи, а сваки има своју вредност интензитета. Да бисмо побољшали слику и користили је за одређене апликације, примењујемо различите операције које су део методе обраде слике.

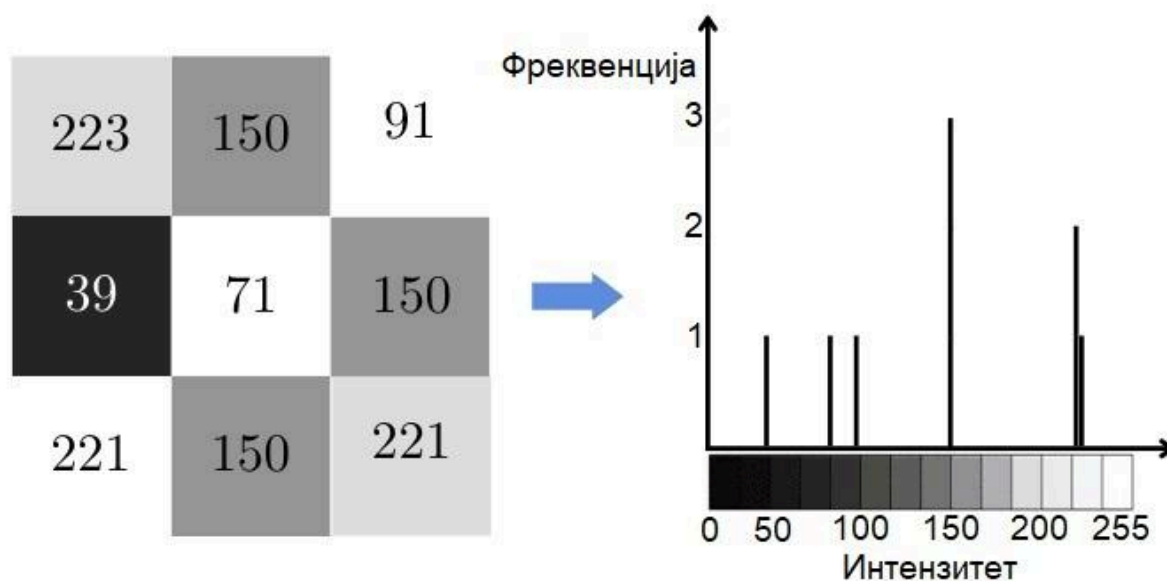
Дигиталне слике се основно деле на три типа а то су монохромне или бинарне слике, слике у сивим тоновима и слике у боји. Вредност пиксела бинарне слике на одређеној локацији  $(x,y)$  обично има вредност 0 за црну или 1 за белу боју. Слике у сивим тоновима имају вредности интензитета у распону од 0 до 255, где 0 представља црну, која постепено прелази до 255, што представља белу боју. Поред тога, колор слике, као што су *RGB* слике, садрже три канала, црвени, зелени и плави. Сваки канал у *RGB* слици има вредности интензитета у распону од 0 до 255.

Операције узорковања и квантизације су део методе обраде слике која претвара континуиране напонске сигнале добијене са сензора у дигиталне слике.



## 2.2. Хистограм слике

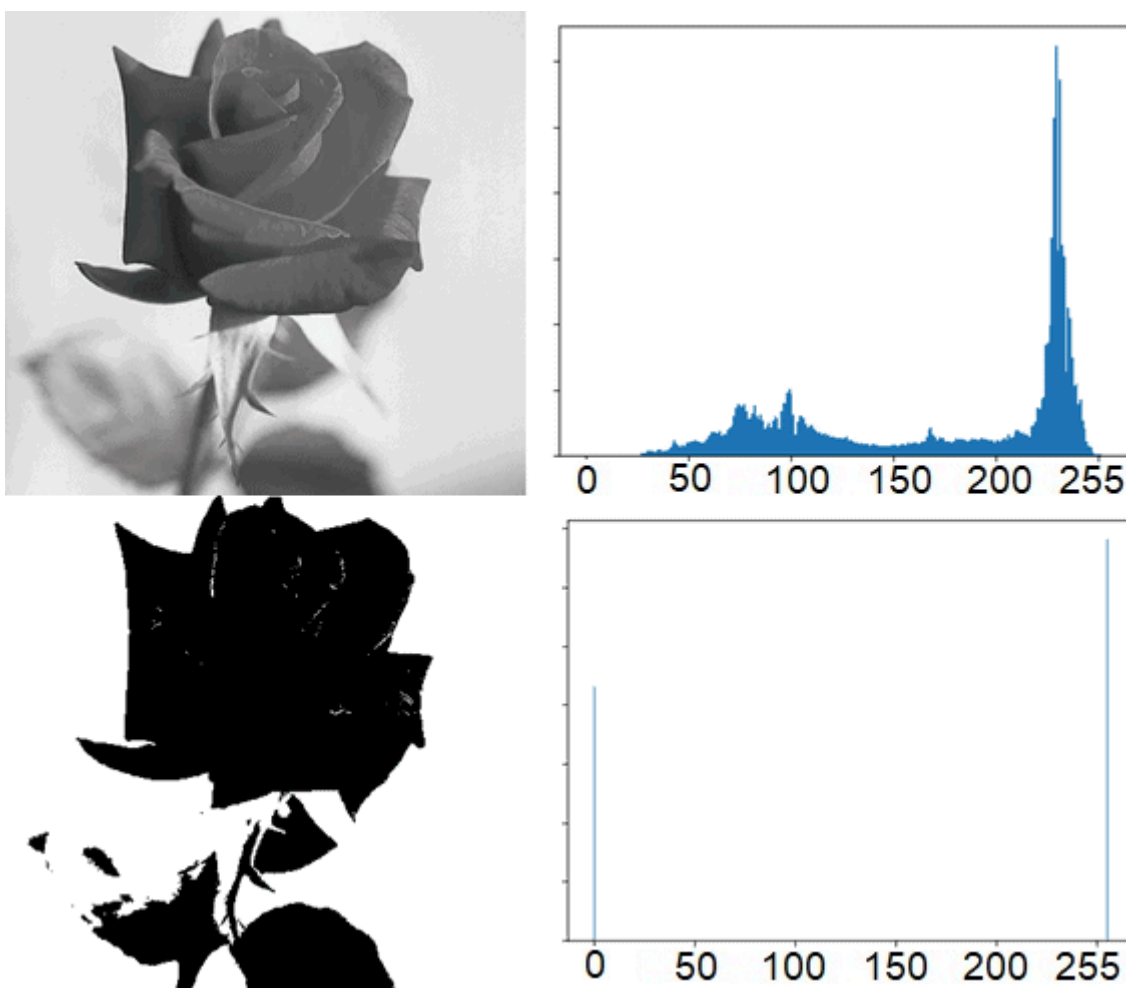
Да би се одредио хистограм слике, потребно је да се преброји колико пута се појављује сваки интензитет. Хистограм ће омогућити да се види колико често се сваки интензитет појављује. У овом примеру на слици 1., интензитет 150 може се видети у три пиксела, због тога ће имати већу фреквенцију у хистограму, висина одговарајуће колоне је 3. Постоје многе области у којима су хистограми корисни, као што су рачунарски вид, обрада слике и фотографија.



Слика 1. Креирање хистограма на основу сиве слике<sup>1</sup>

Хистограми се могу користити за дефинисање прага за сегментацију слике како би се изоловала позадина од објекта. Ако је потребно изоловати ружу на слици број 2. од позадине, може се кренути анализом њеног хистограма. Пошто се види да су већина позадинских пиксела бели или бледо бели. То значи да су већина позадинских пиксела близу 255. Ако се дефинише праг као 156 и узме се сваки пиксел већи од 156 да припада позадини и добије вредност 255, а остали пиксели добију вредност 0, добија се бинарна слика у којој је ружа јасно одвојена. Пошто је измењена слика бинарна (пиксел је или део руже или позадине), нови хистограм има само две могуће вредности. Овај филтер се користити у апликацији овог рада и било би најбоље бинаризовати слику пре извршавања екстракције карактера, најбитније ту је изабрати одговарајући праг.

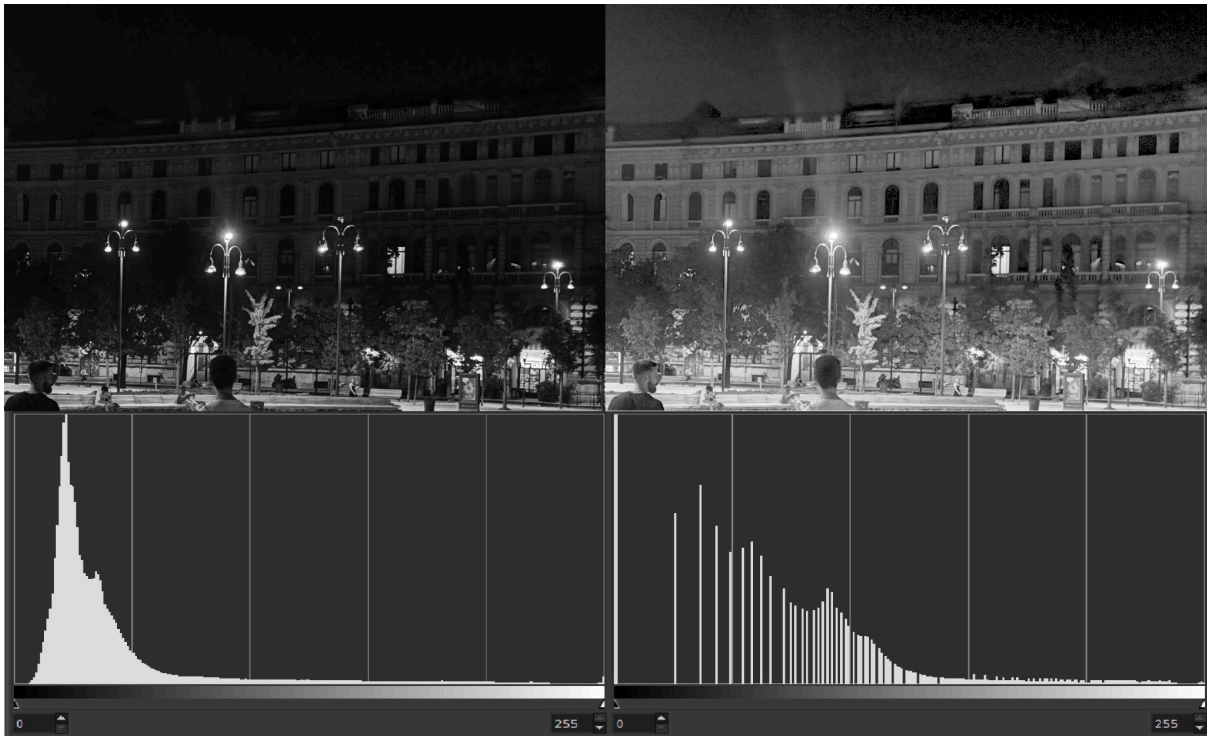
<sup>1</sup> Слике преузете су из чланка *"What Are Image Histograms?"*, [baeldung.com/cs/image-histograms](http://baeldung.com/cs/image-histograms), Saulo Barreto



Слика 2. Слика руже и хистограм пре и после филтера бинаризације<sup>1</sup>

У фотографији користе се хистограми за побољшање слика променом неких њихових својстава. Ово може помоћи да се добију јасније слике или чак лепше фотографије.

На слици 3. приказана је употреба изједначавања хистограма за побољшање контраста слике. Постоји недовољно насељених региона за одређене интензитете, ови региони ће имати више пиксела након обраде, што се у овом случају назива изједначавање. Фотографија на слици 3. је лична фотографија која је имала лошије осветљење и након филтера у *Gimp* алату и помоћу обрада и “растежањем” хистограма је осветљење побољшано. У левом горњем углу је слика сивих тонова без обраде, у десном горњем углу је слика након обраде, испод сваке слике стоји хистограм дате слике, у оса представља број пиксела, а x оса представља вредност пиксела од 0 до 255.



Слика 3. Пример изједначавања хистограма

## 2.3. *Grayscale* филтер

*Grayscale* филтер је техника која се користи за претварање слике у боји у слику у сивим тоновима. Овај филтер уклања боје и задржава само интензитет светлости за сваки пиксел. Резултат је слика у којој су сви пиксели приказани у нијансама сиве, што може олакшати анализу и обраду слике.

Један од честио коришћених метода за претварање слике у боји у сиве тонове је коришћење прорачунатих средњих вредности црвене, зелене и плаве компоненте, али најчешћа формула користи следеће коефицијенте за различите боје, који одражавају људску перцепцију светлости:

$$Y = 0.299 \times R + 0.587 \times G + 0.114 \times B$$

$Y$  је интензитет сиве боје (резултатни пиксел у сивим тоновима),

$R$  је интензитет црвене боје,

$G$  је интензитет зелене боје,

$B$  је интензитет плаве боје.

Ови коефицијенти су одабрани јер људско око најсензитивније реагује на зелену, затим на црвену, а најмање на плаву боју. Користећи ову формулу, сваки пиксел у колор слици се конвертује у одговарајући ниво сиве боје, где  $Y$  представља нову вредност пиксела у сивим тоновима.

## 2.4. Интерполација слике

Интерполација слике је процес прилагођавања димензија слике. То може укључивати повећање или смањење слике и користи се за различите примене као што су уређивање слика, дигитално штампање и *Web* дизајн. Да би се постигла квалитетна интерполација, користе се различити филтери и технике који помажу у очувању што више детаља и минимизирању губитака квалитета. У наставку биће описано пар алгоритама за интерполацију.

*Nearest Neighbor* је најједноставнији метод интерполације слике. Сваком пикселу у новој слици се додељује вредност најближег пиксела из оригиналне слике. Предности су што је брз и једноставан. Недостаци су што може произвести "*blocky*" или пикселизоване ефекте, посебно код повећања слике.

*Bilinear* интерполација користи четири најближа пиксела из оригиналне слике за израчунавање нове вредности пиксела у измењеној слици. Прорачунава просечну вредност пиксела у односу на удаљеност од та четири пиксела. Предности су што пружа глаткији резултат у поређењу са *Nearest neighbor* методом. Недостаци су што може довести до замућења слике.

*Bicubic* интерполација која користи 16 најближих пиксела (у квадратној мрежи) за израчунавање нове вредности пиксела. *Bicubic* интерполација даје бољи квалитет резултата јер узима у обзир више пиксела и користи полиномну функцију за израчунавање интензитета. Предности су што пружа оштрије и глаткије резултате у поређењу са *Bilinear* интерполацијом. Недостаци су што је спорија од *Bilinear* интерполације.



Слика 4. Примери за (a) *Nearest-neighbor*, (b) *Bilinear*, (c) *Bicubic*, (d) Оригинал<sup>2</sup>

<sup>2</sup> Слика преузета из чланка: "*Super-resolution via adaptive combination of color channels*", Jian Xu, Zhiguo Chang, 2015

### 3. Оптичко препознавање карактера

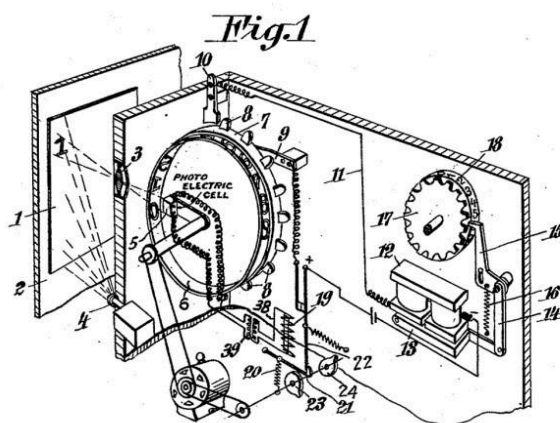
Оптичко препознавање карактера (OCR) је технологија која омогућава конверзију различитих типова докумената, као што су скенирани папирни документи, *PDF* датотеке или слике снимљене камером у претраживе текстуалне податке. OCR користи сложене алгоритме за препознавање и интерпретацију текста унутар слике, омогућавајући корисницима да дигитализују и архивирају документе на ефикасан начин. Ова технологија је посебно корисна у пословним процесима где је потребно брзо и прецизно претраживање великих количина података.

Историја OCR-а почиње крајем 1920-их година када је Густав Таушек (*Gustav Tauschek*) добио патент за свој уређај за оптичко препознавање знакова у Немачкој. Амерички патент додељен је 31. децембра 1935. године за “машину за читање” Густаву Таушеку из Беча, Аустрија, што је био један од најранијих OCR изума. OCR је у почетку коришћен за дигитализацију штампаних текстова и омогућавање њихове машинске читљивости. Како је OCR технологија наставила да напредује, постала је широко коришћена у разним индустријама. Његов уређај је користио механичке шаблоне и фотодетекторе за препознавање карактера. У исто време, у Сједињеним Америчким Државама, Пол Хандел (*Paul Handel*) је развио сличан систем и добио патент 1933. године. Током 1950-их, Дејвид Х. Шепард (*David H. Shepard*) је направио значајан напредак у OCR технологији развијајући уређаје који су могли да конвертују штампани текст у машински језик, што је омогућило даљу обраду података рачунаром.

Dec. 31, 1935.

G. TAUSCHEK  
READING MACHINE  
Filed May 27, 1929

2,026,329



Слика 5. Скица патента Густава Таушека<sup>3</sup>

<sup>3</sup> Слика преузета из “United States Patent Office”, “Reading Machine”, Gustav Tauschek

OCR технологија је значајно напредовала захваљујући развоју вештачке интелигенције и машинског учења. Модерни OCR системи користе напредне алгоритме за препознавање текста, укључујући неуронске мреже које омогућавају високу тачност препознавања чак и код комплексних и нејасних докумената.

### 3.1. OCR са машинским учењем

OCR са машинским учењем је софтвер за оптичко препознавање карактера који користи технологије машинског учења. Примена машинског учења омогућава софтверу да лако разуме и препозна општи контекст документа, кроз скуп правила. У поређењу са традиционалним оптичким препознавањем карактера, OCR са машинским учењем:

- Процесуира и структуриране и неструктуриране податке
- Заснива се на континуираном циклусу учења
- Постиже више од 95% тачности у екстракцији информација

Могућности OCR-а са машинским учењем далеко надмашују било који систем заснован на шаблонима. Са довољно података и правилним методама обуке, OCR са машинским учењем повећава своју тачност и стопе предикције, те може препознати и одредити локацију поља података са високом прецизношћу. На основу тога, софтвер екстрактује информације у складу са сложеношћу документа.

Док машинско учење унапређује OCR у целини, постоје различите подгрупе алгоритама машинског учења и OCR-а које се могу користити за процесе екстракције података. На пример, неки модели се фокусирају на обраду природног језика (*Natural language processing - NLP*) како би ухватили информације из *live chat*-ова, рукописа или *email*-ова. Други модели користе, на пример, дубоко учење.

### 3.2. OCR са дубоким учењем

OCR са дубоким учењем (*Deep learning*) је најкориснији када се ради са несавршено скенираним папирним документима. Оштећене скениране слике могу садржати шум, лоше осветљење или захтевати исправљање нагиба, што отежава екстракцију података за OCR модел без AI технологија. Примена дубоког учења и OCR-а укључује три корака:

- Предпроцесирање слике: AI обрада слике помаже у побољшању квалитета слике, што значи да се шум уклања, осветљење прилагођава и исправља нагиб. Овај први корак је кључан за стварање најбољег окружења за екстракцију података.

- Детекција текста: За лоцирање текста унутар документа може се користити неколико модела. Они креирају оквире око сваког текста идентификованог на слици или документу. Ово је припремни корак за екстракцију података.
- Препознавање текста: Након што је текст лоциран у фајлу, сваки оквир се шаље моделу за препознавање текста. Коначан резултат ових модела је екстрактовани текст из докумената.

Постоји више подтипова алгоритама машинског учења који се могу користити за побољшање квалитета и тачности екстракције података.

### 3.3. OCR алгоритам

OCR алгоритам може се поделити у две различите категорије:

- Алгоритам заснован на шаблонима: Овај алгоритам чита и екстрахује само информације на којима је трениран. Како назив сугерише, обука се врши на унапред дефинисаним шаблонима, што чини овај алгоритам основом традиционалног OCR-а.
- Когнитивни алгоритам: Користи когнитивну обуку и дизајниран је да идентификује сваки карактер истовремено. Овај систем скенира цео документ и гради просторну “мапу” локације важних података.

Когнитивни алгоритам функционише тако што OCR систем скенира област документа или слике да би идентификовао оно што претпоставља да је скуп карактера. Систем затим додељује вредност “0” или “1” сваком пикселу у непосредној близини карактера, на основу тога да ли је боја црна или бела. Све вредности означене са “1” су црне и показују облик сваког карактера. OCR модел затим упоређује почетну просторну “мапу” коју је створио током фазе претходне обуке и тачно идентификује који карактери чине податке. Поред самих карактера, најефикаснији OCR алгоритам ослања се на контекстуалне трагове и просторни распоред података. Обимна тестирања доследно показују да овај тип модела надмашује конвенционалне OCR моделе у погледу ефикасности и тачности.

### 3.4. Како OCR ради

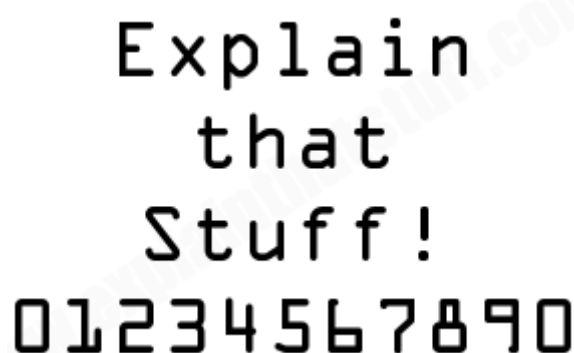
Различите верзије слова "А", штампане у различитим компјутерским фонтовима, показују значајну варијацију, али деле основне сличности, скоро све су формиране од две угаоне линије које се састају на врху и једне хоризонталне линије између њих. OCR систем препознаје ове основне облике, упркос разликама у фонтовима или стиловима писања. Генерално, OCR системи користе два основна приступа за препознавање знакова:

1. Препознавање шаблона: Упореджује цео знак са унапред дефинисаним шаблоном карактера.
2. Детекција карактеристика: Препознаје основне линије и потезе из којих су карактери сачињени, омогућавајући идентификацију и препознавање.

#### 3.4.1. Препознавање узорака

Препознавање шаблона функционише тако што упоређује скенирани знак са унапред дефинисаним шаблоном карактера. Да би овај метод био успешан, сви знакови морају бити стандардизовани.

У 1960-им годинама развијен је посебан фонт, OCR фонт, како би се олакшало препознавање текста на банковним чековима и другим документима. Свако слово у овом фонту је имало исту ширину, а потези су дизајнирани тако да се свако слово може лако разликовати од свих других. OCR системи су тренирани да препознају овај фонт, што је поједноставило процес препознавања. Међутим, OCR-A font се користи само у специфичним случајевима, док већина штампаног текста користи различите фонтове, што чини OCR сложенијим задатком у реалним сценаријима.



Explain  
that  
Stuff!  
01234567890

Слика 6. OCR фонт<sup>4</sup>

---

<sup>4</sup> Сlike. преузете су из чланка "Optical character recognition (OCR)", <https://www.explainthatstuff.com/how-ocr-works.html>, Chris Woodford, 2023

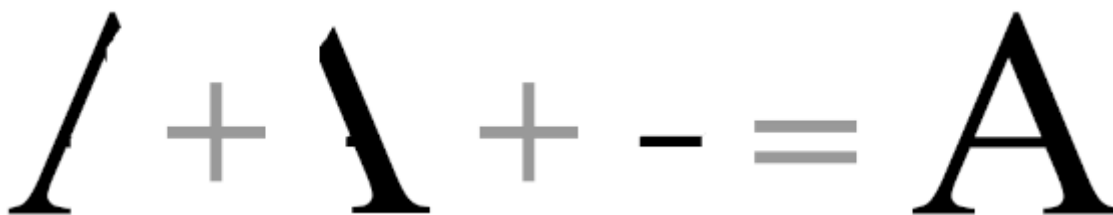


### 3.4.2. Детекција карактеристика

Такође познат као детекција карактеристика или интелигентно препознавање карактера (*Intelligent character recognition - ICR*) је много напреднији начин препознавања карактера.

Ако се виде две угаоне линије које се састају у тачки на врху, у центру, а између њих постоји хоризонтална линија отприлике на пола пута доле, то је слово А. То се може применити у препознавању већину великих слова А, без обзира којим фонтом су написана. Уместо да се препозна комплетан образац слова А, детектују се карактеристике појединачних компоненти (угаоне линије, укрштене линије или било шта друго) од којих је слово направљено.

Већина модерних *OCR* програма (они који могу да препознају штампани текст у било ком фонту) ради детекцијом карактеристика, а не препознавањем шаблона. Неки користе неуронске мреже (рачунарске програме који аутоматски издвајају обрасце на начин сличан мозгу).



Слика 7. Откривање карактеристика карактера<sup>4</sup>

### 3.5. Како се ради са *OCR* алатом у пракси

1. Штампаче: Потребно је добити најбољи могући одштампани документ вашег постојећег документа. Квалитет оригиналног отиска чини велику разлику у тачности *OCR* процеса. Прљави трагови, набори, мрље од кафе, мрље од мастила и било који други залутали трагови ће смањити вероватноћу исправног препознавања слова и речи.

2. Скенирање: Испис пролази кроз оптички скенер. Скенери са увлачењем листова су бољи за *OCR* од равних скенера јер можете скенирати странице једну за другом. Већина модерних *OCR* програма ће скенирати сваку страницу, препознати текст на њој, а затим аутоматски скенирати следећу страницу. Дobar дигитални фотоапарат или камера телефона могу да креирају одличне фотографије страница, могуће је искористити и апликације за паметне телефоне које служе баш за скенирање докумената.

3. Црно-бела слика: Прва фаза у *OCR*-у укључује генерисање црно-беле (двобојне/једнобитне) верзије скениране странице у боји или нијансама сиве, слично ономе што бисте видели како излази из факс машине. *OCR* је у суштини бинарни процес, препознаје ствари које постоје или не. Ако је оригинална скенирана слика савршена, свака црна која садржи биће део карактера који треба препознати, док ће свака бела бити део позадине. Свођење слике на црно-белу је стога прва фаза у откривању текста који треба да се обради иако може да унесе и грешке.

4. *OCR*: Свих *OCR* програми се мало разликују, али генерално обрађују слику сваке странице тако што препознају текст знак по знак, реч по реч и ред по ред. Средином 1990-их, *OCR* програми су били толико спори да је могуће гледати их како читају реч по реч и обрађују текст док се чека, рачунари су сада далеко бржи и *OCR* даје резултате прилично брзо.

5. Основна исправка грешака: Неки програми дају прилику да се прегледа и исправи свака реч, они тренутно обрађују целу страницу, а затим користе уграђену проверу правописа да истакну све очигледно погрешно написане речи које могу указивати на погрешно препознавање, тако да можете аутоматски да исправите грешке. Напреднији *OCR* програми имају додатне функције за проверу грешака које помажу да се уоче грешке.

6. Анализа распореда: Добри *OCR* програми аутоматски откривају сложене изгледе страница, као што су више колоне текста, таблице, слике итд. Слике се аутоматски претварају у графике, таблице се претварају у таблице, а колоне су правилно подељене.

7. Лектура: Чак ни најбољи *OCR* програми нису савршени, посебно када раде са веома старим документима или лошег квалитета штампаног текста. Зато завршна фаза у *OCR*-у увек треба да буде добра, старомодна људска провера, односно лектура.

## 4. Технологије и библиотеке

### 4.1. *Tesseract*

*Tesseract* је *OCR* (*Optical Character Recognition*) механизам који се користи за препознавање и конверзију текста са слика у машински кодирани текст. *Tesseract* је почео као *PhD* пројекат у сарадњи са *HP* и Бристолом и добио је примену као надоградња на *HP*-овим скенерима. *Tesseract* је *OCR* алат је *HP* развијао између 1984. и 1994. године. Крајем 2005. године, *HP* је објавио *Tesseract* као *open-source* пројекат.

*Tesseract* очекује да је његов улаз бинарна слика са опционално дефинисаним полигоналним регионима текста. Обрада следи традиционални корак-по-корак процес, али су неке фазе биле необичне за своје време, а могуће и сада. Први корак је анализа повезаних компоненти у којој се чувају обриси компоненти. *Tesseract* је вероватно био први *OCR* систем који је могао тако лако да обрађује бело-на-црно текст. У овој фази, делови се групишу у блобове. Блобови се затим организују у линије текста, а линије и региони се анализирају за фиксну ширину слова или пропорционални текст. Линије текста се деле на речи различито у зависности од врсте размаке између карактера. Текст са фиксном ширином слова се одмах дели по ћелијама карактера. Пропорционални текст се дели на речи користећи јасне и нејасне размаке.

Препознавање се затим наставља као процес у два пролаза. У првом пролазу, покушава се препознавање сваке речи редом. Свака реч која је задовољавајућа користи се као тренинг податак за адаптивни класификатор. Адаптивни класификатор тада добија прилику да тачније препозна текст ниже на страници.

Пошто је могуће да адаптивни класификатор научи нешто корисно прекасно да би допринео при врху странице, покреће се други пролаз преко странице, у којем се поново препознају речи које нису биле довољно добро препознате. Коначна фаза решава нејасне размаке и проверава алтернативне претпоставке за *x*-висину да би се локализовао текст са малим словима. У следећим деловима биће описано пар метода које *Tesseract* користи.

#### 4.1.1. Уклапање основне линије

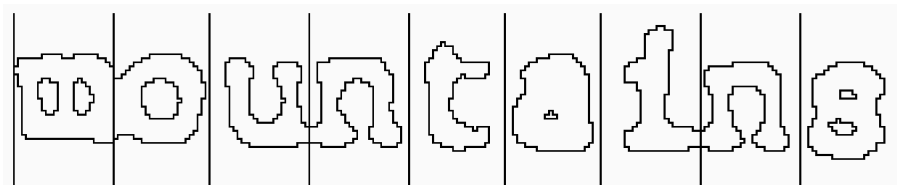
*Tesseract* користи алгоритам за проналажење линија који препознаје искошене странице без губитка квалитета, филтрира делове слике (блобове), и прецизно уклапа основне линије, што омогућава обраду закривљених линија текста. Слика 9. представља начин извлачења линија како би се утврдила закривљеност текста.

Volume 69, pages 872-879.

Слика 8. Пример обележеног закривљеног текста<sup>5</sup>

#### 4.1.2. Детекција фиксне ширине слова и одсецање

Детекција фиксне ширине слова и одсецање је принцип рада где се анализирају линије као на слици 9., да би се одредило да ли имају константну ширину слова. Када наиђе на такав текст, деле се речи на слова користећи детектовану ширину и олакшава даље кораке у делу за препознавање речи.



Слика 9. Пример детекције фиксне ширине слова<sup>5</sup>

#### 4.1.3. Променљива ширина у тексту

Текст са променљивим размаком представља проблем у OCR системима. На слици 10. је приказан пример текста са различитом дистанцом између слова. Не постоји хоризонтални размак између речи „of“ и „financial“. *Tesseract* решава овај проблем тако што мери размак ограниченог вертикалног распона основне и средње линије. Размаци који су близу границе се обележавају као нејасни и коначна одлука се доноси након корака препознавања речи.

of 9.5% annually while the Fed-  
erated junk fund returned 11.9%  
*fear of financial collapse,*

Fig. 3. Some difficult word spacing.

Слика 10. Текст са променљивим размаком између речи<sup>5</sup>

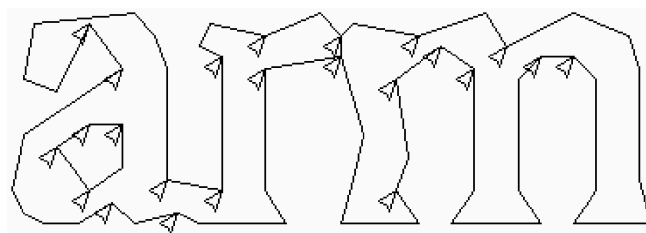
#### 4.1.4. Препознавање речи

Део процеса свих система за препознавање карактера је како идентификовати реч коју треба сегментирати у карактере. Иницијална сегментација излаза линија се прво класификује. Остатак препознавања речи се примењује само на текст са променљивом ширином слова.

<sup>5</sup> Слике преузете из чланка: "An Overview of the Tesseract OCR Engine", Ray Smith, Google Inc.

#### 4.1.5. Одсецање повезаних карактера

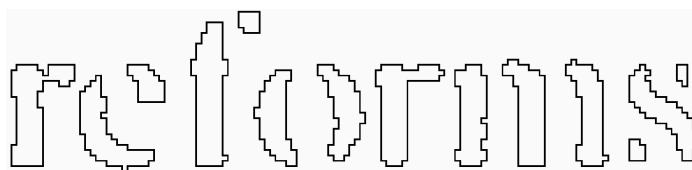
Док су резултати незадовољавајући, Tesseract покушава да побољша резултат одсецањем речи са најмање додељеном прецизношћу из класификатора карактера. Тачке које ће се користити за одсецање се проналазе у полигоналној апроксимацији. Потребно је до 3 пара тачака за успешно одсецање из ASCII сета. Одсецања се врше у редоследу приоритета. Тачке које не унапређују резултат се одбацују, али остају како би могле да се искористе у асоцијатору уколико је то потребно. На слици 11. су обележене могуће тачке за одсецање.



Слика 11. Могуће тачке за одсецање<sup>5</sup>

#### 4.1.6. Повезивање изломљених карактера

Уколико и након одсецања реч није довољно добра, шаље се асоцијатору. Tesseract овде врши A\* претрагу графа сегментације, да би дошао до могућих комбинација одсечених блобов. Ово се врши без израде графа, преко hash табеле обилажених стања. A\* претрага наставља тако што проналази одговарајуће кандидате из приоритетног реда и евалуира их док не нађе приближно добро решење. На слици 12. налази се пример изломљених карактера.



Слика 12. Изломљени карактери<sup>5</sup>

#### 4.1.7. Слични OCR алати

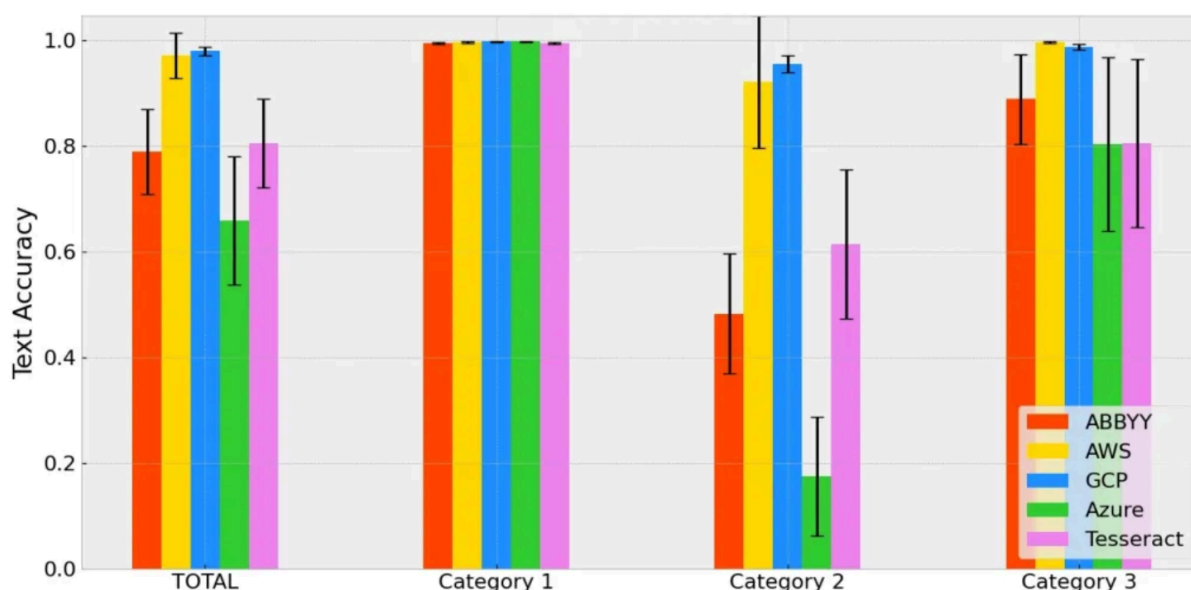
Abbyy FineReader је комерцијални OCR алат познат по високом квалитету и прецизности. Пружа решења за различите потребе, укључујући конверзију текста из слика и PDF-а у уређиве формате. Цене могу варирати, али обично се крећу од 200 до 500 евра у зависности од верзије и лиценце.

Amazon Textract је OCR услуга из AWS-а која аутоматски извлачи текст и податке из докумената. Трошкови зависе од количине обрађених страница.

Основне цене почињу од неколико центи по страници, али се могу повећати у зависности од коришћених функција.

*Google Cloud Vision (GCP)* пружа OCR услугу као део *Google Cloud* платформи. Цене су засноване на количини обрађених слика. Основне цене су око 1.50 USD за 1000 слика у стандардној верзији, али могу да варирају у зависности од конкретних услуга и количине података.

*Microsoft Azure Computer Vision* укључује OCR функцију која омогућава извлачење текста из слика. Цена зависи од броја обрађених слика и коришћених функција. Основне цене су око 1 USD за 1000 слика, али могу се променити у зависности од пакета и обима коришћења.



Слика 13. График са поређењем различитих OCR алата<sup>6</sup>

На слици 13. на којој је приказано поређење *Tesseract*-а са сличним алатима види се да овај алат много не заостаје за комерцијалном конкуренцијом на тржишту. *Tesseract OCR* је изабран због неколико кључних предности које га издвајају у односу на друге алате:

1. Отворени код: *Tesseract* је потпуно бесплатан и отвореног кода, што значи да можете лако прилагодити и проширити његове функционалности према специфичним потребама пројекта.
2. Флексибилност: Може се комбиновати са алатима као што су *OpenCV* или друге библиотеке за обраду слике, што омогућава унапређење препроцесирања слике и побољшање тачности препознавања. Може се користити на било којој платформи.
3. Подршка за више језика: *Tesseract* подржава преко 100 језика, што га чини погодним за пројекте на глобалном нивоу, без потребе за додатним језичким пакетима.

<sup>6</sup> Слика преузета из "OCR in 2024: Benchmarking Text Extraction/Capture Accuracy", [research.aimultiple.com/ocr-accuracy](https://research.aimultiple.com/ocr-accuracy/), Cem Dilmegani

4. Активна заједница и стална ажурирања: Као део *Google* екосистема, *Tesseract* има велику подршку и константно добија унапређења од стране заједнице и развијача.
5. Интеграција са *.NET*: *Tesseract* се може лако интегрисати у *C#* пројекте, пружајући флексибилно и прецизно решење за препознавање текста.
6. Бесплатан: За разлику од поменутих технологија, *Tesseract* не захтева никакве новчане издатке у виду плаћања по броју одрађених страница и претплата.

Због ових карактеристика, *Tesseract* је био избор јер пружа висок степен прилагодљивости, подржава различите језике и има добру тачност у препознавању текста. Осим тога, због отвореног кода, могуће је модификовати га према специфичним захтевима без потребе за великим улагањима у комерцијална решења.

## 4.2. Emgu.CV

*Emgu.CV* је *cross-platform*ска *.NET* библиотека (*wrapper*) за *OpenCV*, која омогућава једноставну интеграцију *OpenCV* функционалности у *.NET* апликације користећи језике као што су *C#* и *VB.NET*. Ова библиотека нуди богат скуп функција за обраду слика, од којих су многе кључне за претходну обраду слика. Које је потребно обавити пре него што се проследе *OCR* системима за препознавање текста. Једна од најчешће коришћених функција у оквиру ове библиотеке је бинаризација и *thresholding*, које се користе за побољшање квалитета слике и повећање тачности препознавања карактера.

Бинаризација и *thresholding* су основне технике у обради слика које се користе за конверзију сиве слике у бинарну слику, при чему се пиксели деле на беле и црне вредности на основу одређеног прага. Ове технике су посебно корисне у *OCR* апликацијама, јер помажу у уклањању шума и повећавају контраст између текста и позадине, што олакшава препознавање карактера.

*Emgu.CV* пружа неколико метода за бинаризацију и *thresholding*, укључујући глобални *thresholding*, адаптивни *thresholding* и *Otsu* метод. У наставку су неке од примена ових метода:

Глобални *thresholding* примењује један праг за целу слику. Сви пиксели испод прага се постављају на црно, док се сви пиксели изнад прага постављају на бело.

```
Mat src = CvInvoke.Imread("input_image.png", ImreadModes.Grayscale);
Mat dest = new Mat();
double threshValue = 128.0;
CvInvoke.Threshold(src, dest, threshValue, 255, ThresholdType.Binary);
```

Листинг 1. Глобални *thresholding*

Адаптивни *thresholding* користи различите прагове за различите делове слике, што је корисно када слика има променљиву осветљеност.

```
Mat src = CvInvoke.Imread("input_image.png", ImreadModes.Grayscale);
Mat dest = new Mat();
CvInvoke.AdaptiveThreshold(src, dest, 255, AdaptiveThresholdType.MeanC, ThresholdType.Binary, 11, 2);
```

Листинг 2. Адаптивни *thresholding*

*Otsu* метод аутоматски израчунава оптимални праг за бинаризацију слике на основу хистограма пиксел вредности.



```
Mat src = CvInvoke.Imread("input_image.png", ImreadModes.Grayscale);  
Mat dest = new Mat();  
CvInvoke.Threshold(src, dest, 0, 255, ThresholdType.Binary | ThresholdType.Otsu);
```

Листинг 3. *Otsu* метод

У оквиру ове апликације, *Emgu.CV* се користи за претходну обраду улазних слика како би се побољшала тачност *OCR* препознавања. Примена бинаризације и *thresholding*-а омогућава јаснији контраст између карактера и позадине, што резултира бољим препознавањем карактера и тачнијим генерисањем фонтова.

### 4.3. *Windows Forms*

*Windows Forms* представљају део *.NET Framework*-а који омогућава развој богатих графичких корисничких интерфејса (*Graphical user interface - GUI*) за *Windows* апликације. Као развојна платформа, *Windows Forms* нуде низ предности које их чине одличним избором за израду *desktop* апликација, посебно у контексту апликација које захтевају сложене интеракције корисника, као што је ова апликација. Предности коришћења *Windows Forms* су:

1. Једноставност и продуктивност: *Windows Forms* омогућавају брзо и једноставно креирање *GUI* елемената помоћу *drag-and-drop* визуелног дизајнера у *Visual Studio* окружењу. Овај приступ омогућава програмерима да брзо развију и итеративно унапређују дизајн корисничког интерфејса, повећавајући продуктивност и скраћујући време развоја.
2. Богата библиотека контрола: *Windows Forms* долазе са богатим скупом уграђених контрола, као што су дугме, текстуална поља, листе, табеле, менији и многе друге. Ове контроле могу се лако прилагодити и комбиновати како би се креирао функционалан и атрактиван кориснички интерфејс.
3. Добра подршка за *.NET Framework*: Као део *.NET Framework*-а, *Windows Forms* имају приступ свим погодностима овог окружења, укључујући робусну библиотеку класа, подршку за више језика (*C#, VB.NET, F#*), и напредне функције као што су мрежна комуникација, рад са базама података и обрада слика.
4. Компатибилност и стабилност: *Windows Forms* су већ дуго присутне и доказале су се као стабилна платформа за развој апликација. Компатибилност са старијим верзијама *Windows* оперативног система чини их поузданим избором за широк спектар корисника.
5. Визуелни дизајнер: *Visual Studio* нуди напредни визуелни дизајнер за *Windows Forms*, омогућавајући програмерима да интуитивно распоређају

и конфигуришу контроле на форми. Ово чини развој интерфејса брзим и једноставним, без потребе за ручним писањем великих количина кода.

Избор *Windows Forms* за развој апликације омогућава једноставно креирање богатих и интерактивних корисничких интерфејса, уз искоришћавање предности *.NET Framework*-а.

## 4.4. Верзије и покретање

Комплетна апликација је реализована помоћу *.NET* окружења односно *framework*-а. Ово окружење је одабрано зато што поседује велику подршку за обраду, приказ и рад са сликама. Апликација има пет основне *Windows* форме за приказ на којима се налазе све остале контроле. Верзија окружења коришћеног у апликацији је 4.7.2.

```
<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
```

Листинг 4. Верзија *.NET* Окружења

Главне библиотеке које су биле неопходне за реализацију пројекта су библиотека *Tesseract* и *Emgu.CV*. *Tesseract* библиотека се користи као OCR за препознавање текста, карактера, проналазак позиције карактера итд. *Emgu.CV* је библиотека која у основи користи *OpenCV* и користи се за различите обраде слика у апликацији, као што је на пример бинаризација слике.

```
<package id="EMGU.CV" version="4.1.1.3497" targetFramework="net472" />  
<package id="Tesseract" version="3.3.0" targetFramework="net472" />
```

Листинг 5. Верзије главних библиотека

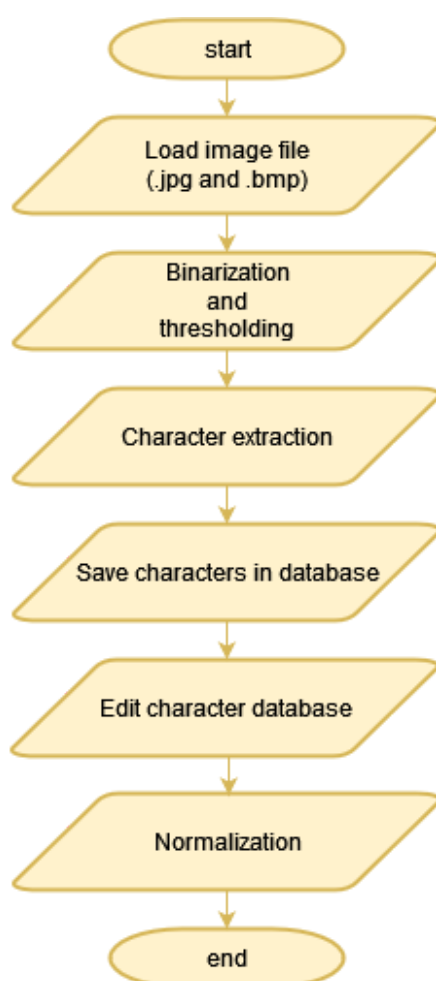
За покретање апликације и за даљи њен развој довољно би било отворити је у *Visual Studij*-у и покренути. Команде у конзоли за *build* и покретање су *dotnet build* и *dotnet run*, ове команде треба да додају све пакете и покрену апликацију. У случају да дође до неке грешке могуће је додати пакете са неком од наредних команди *nuget restore* и *dotnet restore*. Уколико корисник жели да користи апликацију потребно је само покренути инсталацију и пратити кораке инсталације и апликација ће бити спремна за употребу. Тестирана је на оперативним системима *Windows 7* и *10*.

## 5. Дијаграми апликације

У овом делу рада су приказани дијаграми најважнијих делова функционалности и архитектуре апликације, главна њихова сврха је олакшати наставак развоја апликације кроз боље разумевање структуре система.

### 5.1. Дијаграм тока комплетног процеса обраде

Циљ апликације овог рада је био да обухвати цео процес потребан да се изведе прецизно препознавање, обрада и сортирање карактера. (Слика 14.)



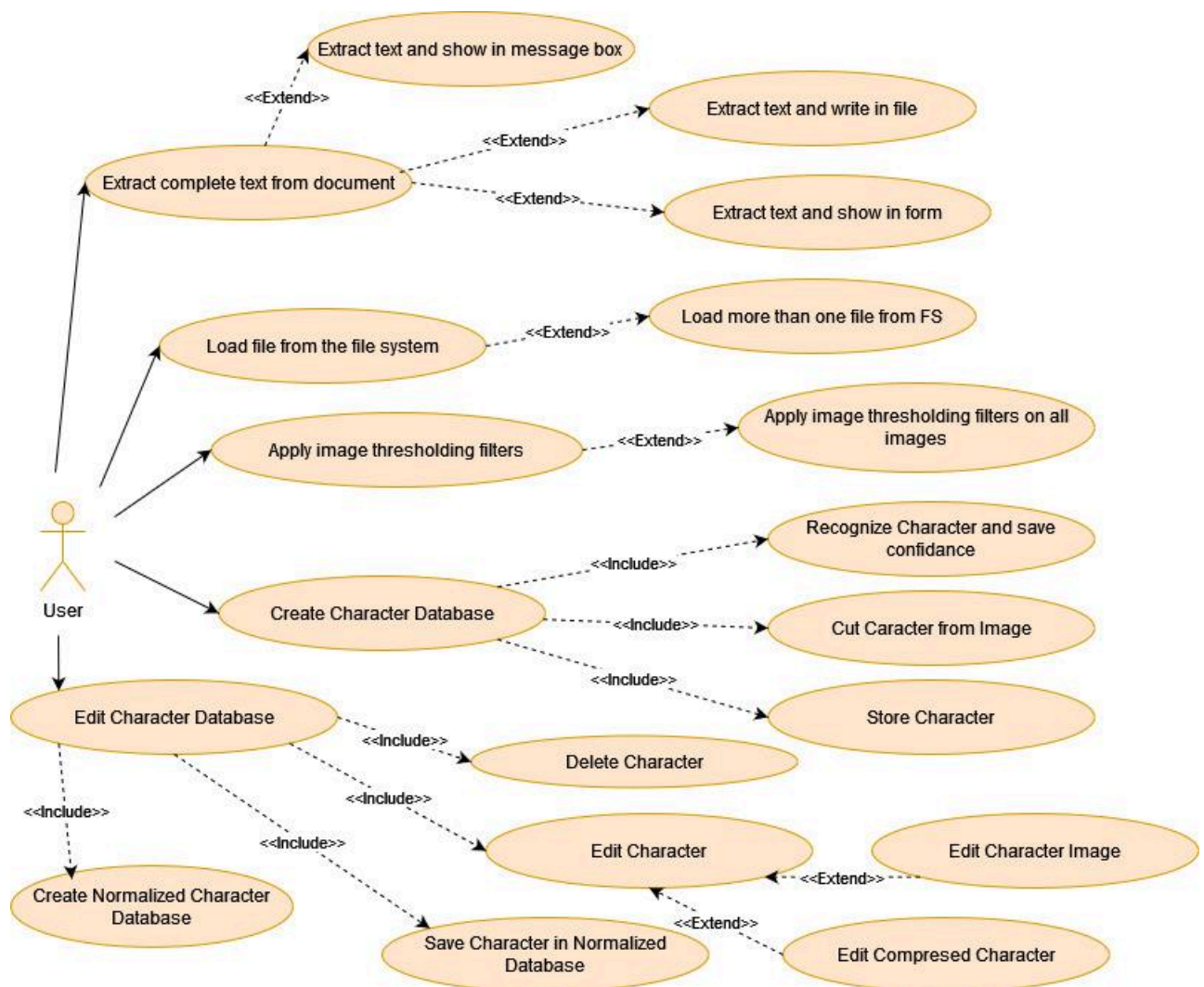
Слика 14. Дијаграм тока

Ток процеса креће са читавањем скенираног документа (Load image file), затим наставља са обрадом од које је потребно добити бинаризовану слику (Binarization and thresholding). Када се прође кроз почетну обраду слике наставља се кроз методе *Tesseract* библиотеке које извлаче (Character extraction) и чувају све препознате карактере заједно са параметром о вероватноћи да је то ај карактер у бази (Save characters in database). Након

екстракције постоји могућност додатне обраде појединачних карактера (Edit character database), а последња операција је нормализација базе података (Normalization) која чува најбоље представнике сваког карактера у јединствену вазу.

## 5.2. Дијаграм случајева коришћења

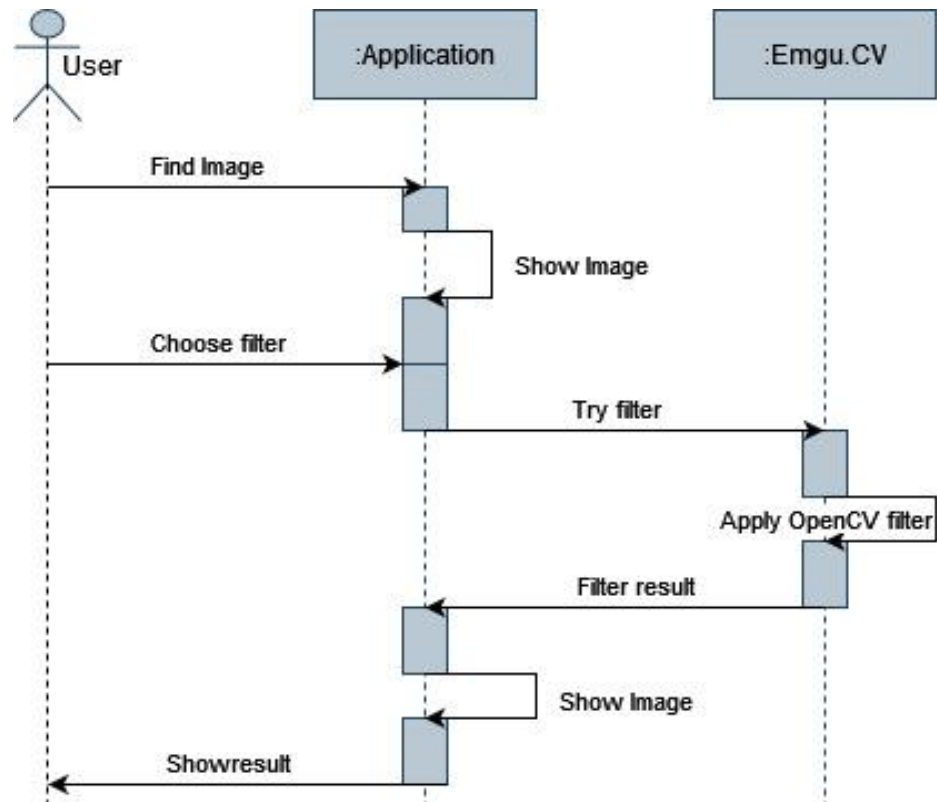
На слици 15. приказан је Дијаграм случајева коришћења целокупног система апликације.



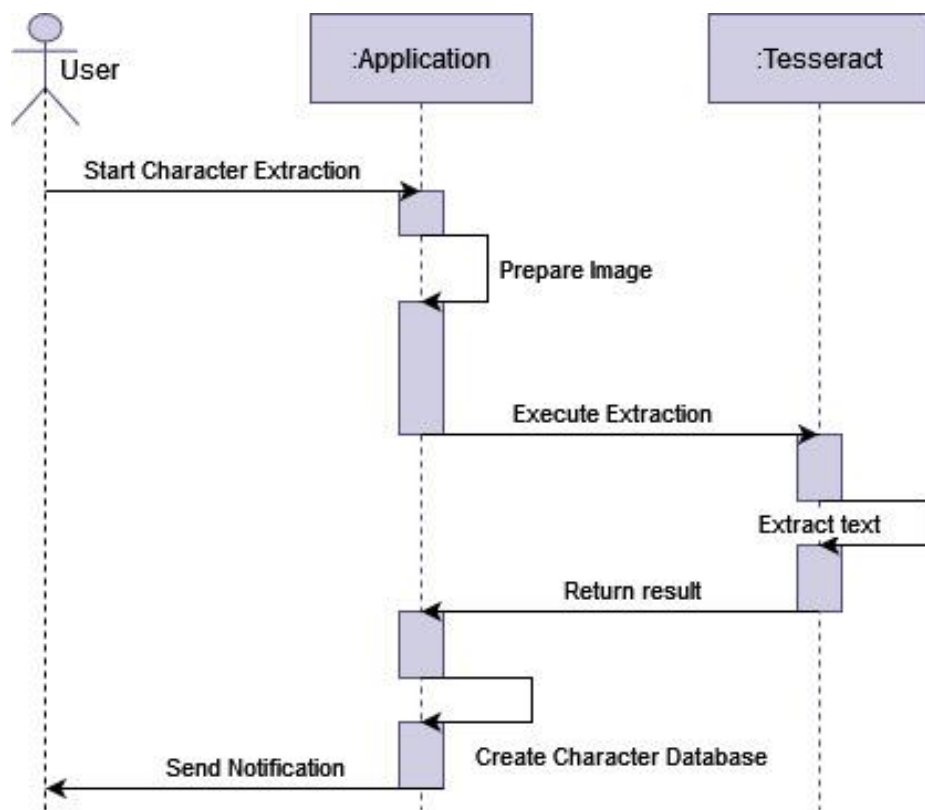
Слика 15. Дијаграм случајева коришћења

### 5.3. Секвенцијални дијаграми

На сликама 16. и 17. приказани су секвенцијални дијаграми две главне обраде ове апликације. Први на слици 16. представља ток команди, обраду и обавештења између корисника, апликације и библиотеке *Emgu.CV*. Други на слици 17. представља секвенцијални дијаграм екстракције карактера у коме учествује библиотека *Tesseract*.

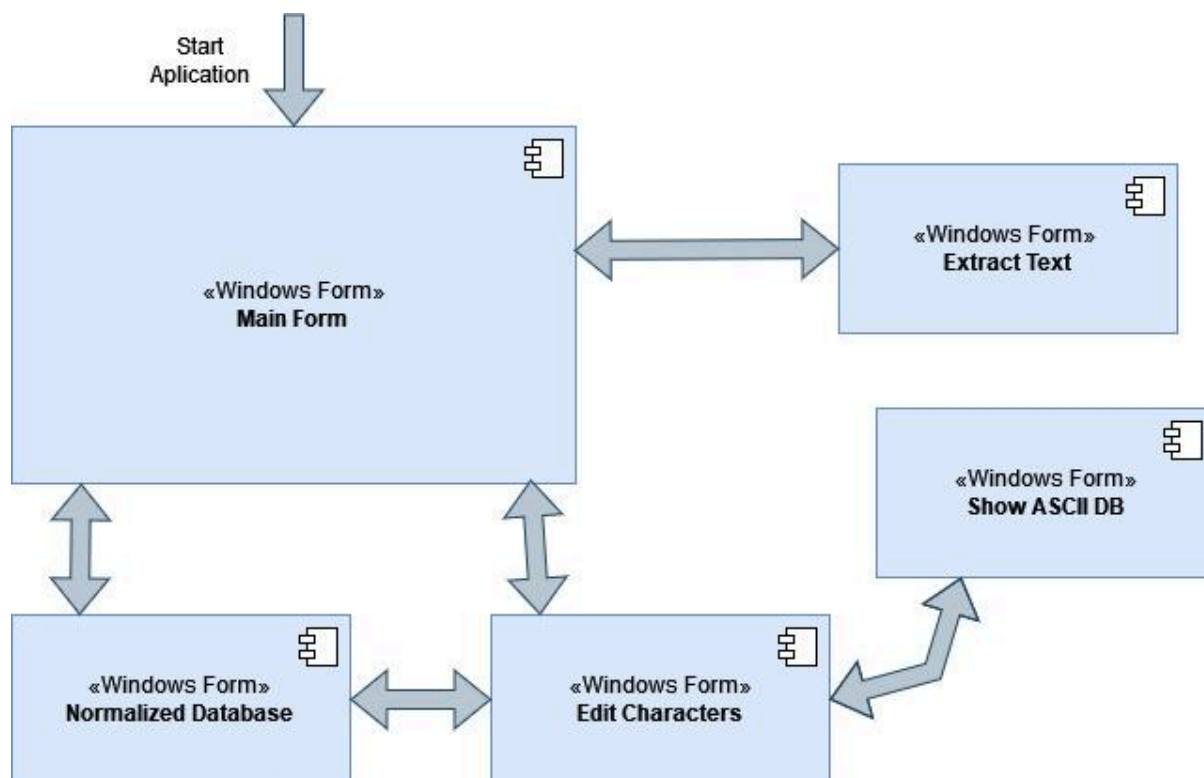


Слика 16. Секвенцијални дијаграм примене филтера над сликама



Слика 17. Секвенцијални дијаграм екстракције карактера

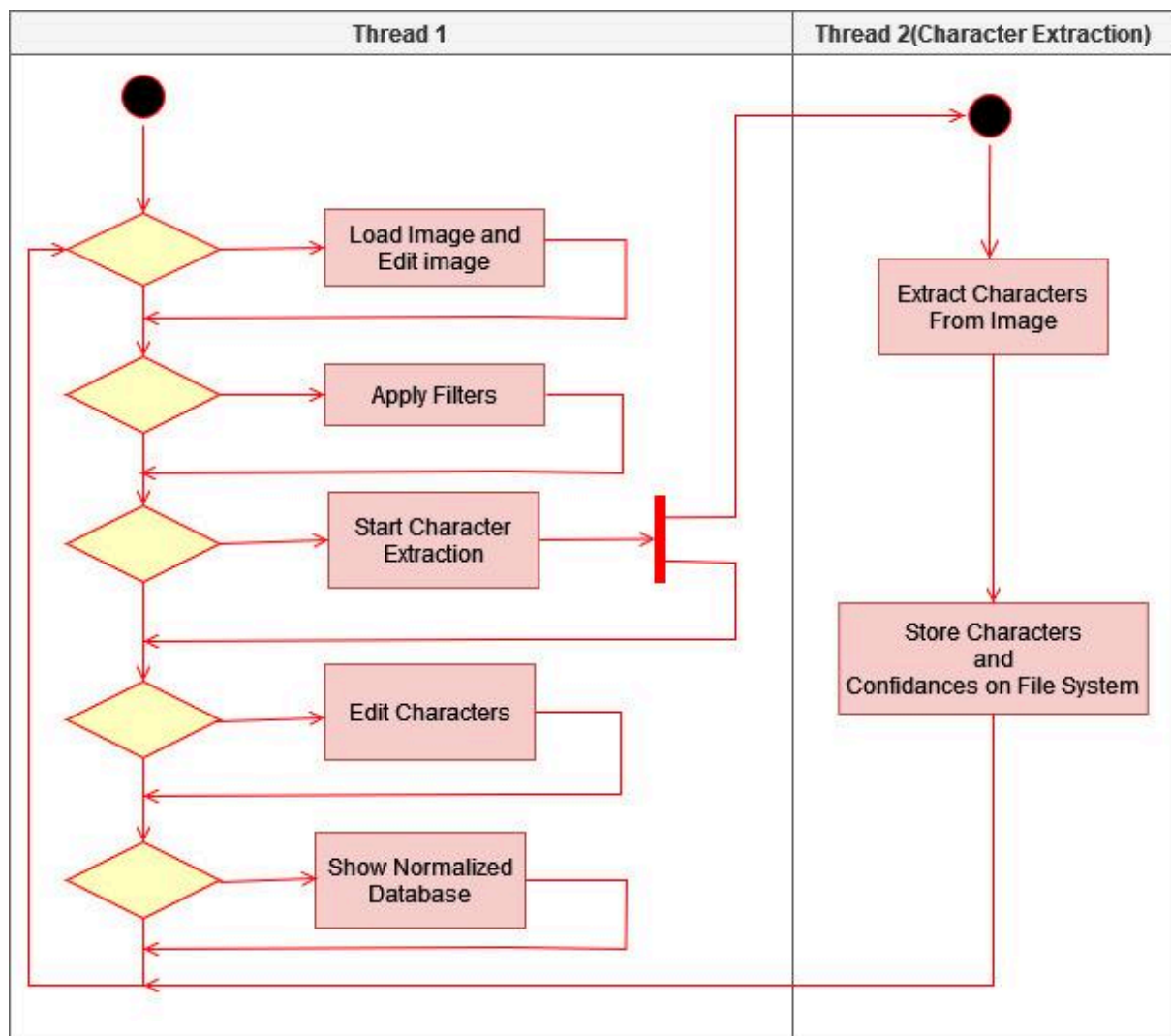
## 5.4. Дијаграм *Windows Form* компоненти



Слика 18. Дијаграм *Windows Form* компоненти

Дијаграм на слици 18. представља дијаграм свих могућих креираних *Windows Form* компоненти из апликације овог рада. Највећа компонента представља главну форму апликације од које почињу почетне обраде од учитавања скенираних докумената, до екстракције текстуалних елемената. Стрелице између сваке од компоненти представљају ток којим се може пролазити из једне у другу *Windows* форму. “*Edit Characters*” форма представља форму у којој се након сепарације карактера врши обрада појединачних карактера и извршава нормализација. Из ове форме је могуће стићи у форме “*Normalized Database*” и “*Show ASCII DB*” које служе за приказ резултата након комплетног процеса. “*Extract Text*” форма је додатна форма која служи за основно препознавање текста из целокупног документа, то је додатна функционалност и неће бити описивана у овом раду.

## 5.5. Дијаграм активности

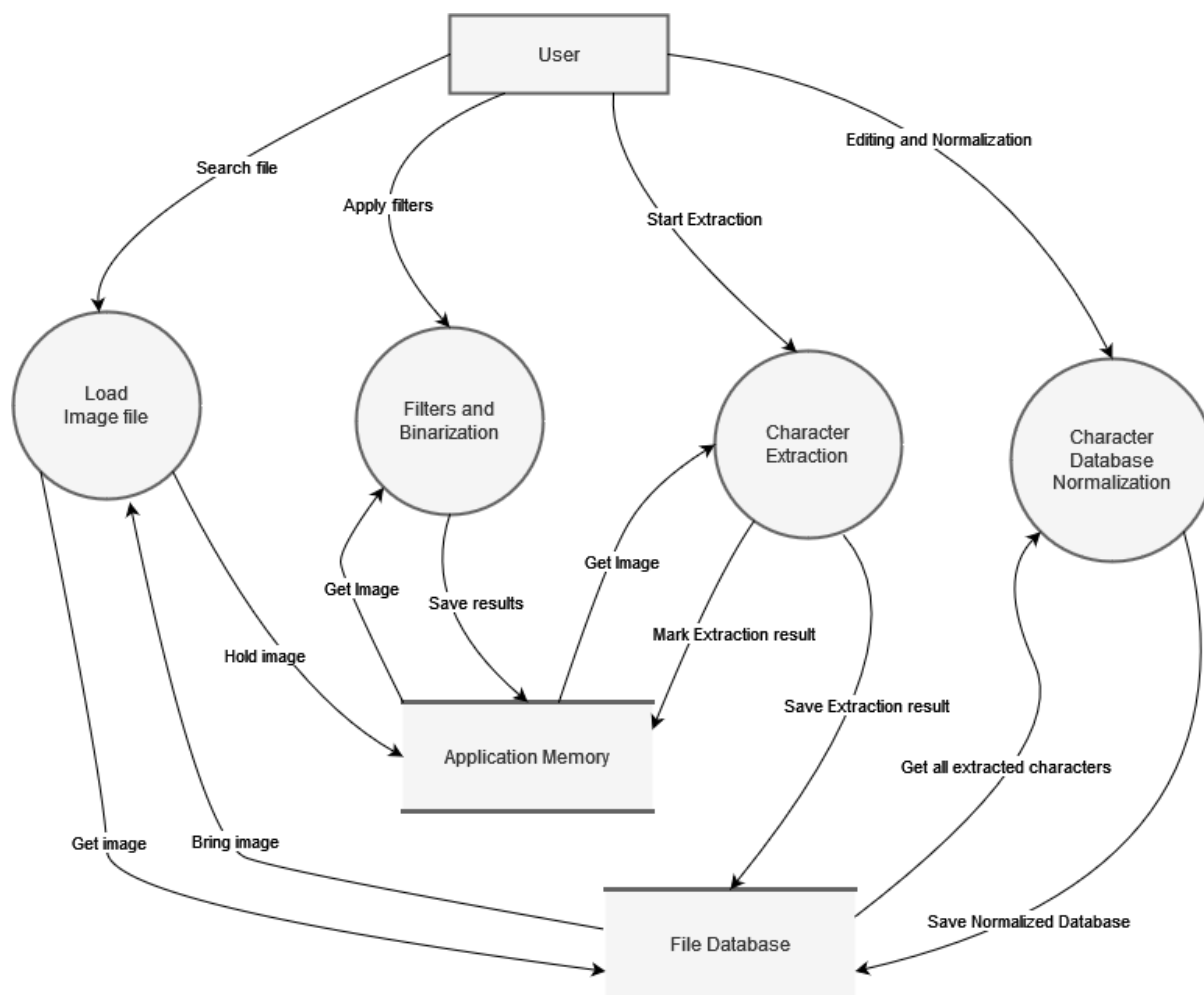


Слика 19. Дијаграм активности паралелних радњи апликације

Слика 19. представља дијаграм активности *Windows Form* апликације. Као што се може видети, дијаграм је подељен на два дела, која представљају засебне паралелне обраде. Обрада екстракције карактера и чување карактера у бази на фајл систему је због трајања обраде, како не би блокирала рад апликације и како корисник не би чекао док је апликација замрзнута, извршава се као паралелна обрада у односу на остатак апликације.



## 5.6. Дијаграм протока података



Слика 20. Дијаграм протока података

На слици 21. представљен је дијаграм протока података (*Data Flow*). Корсник (*User*) је представљен као компонента од које крећу улазне команде, а операције обележене круговима представљају функционалне целине кроз које се подаци обрађују. Две компоненте “*Application Memory*” и “*File Database*” представљају целине од којих подаци полазе и у које се враћају након обраде, а линије између функционалних делова и ових компоненти представљају те токове података.

## 6. Имплементација

За имплементацију ове апликације одабрано је окружење *Windows Forms* уз које се користи језик *C#* који има велику подршку за битмапе(*Bitmap*), за операције са сликама фајловима и велики број библиотека везаних за обраду слике. У овом поглављу су описани основни делови кључних целина везани за комплетну припрему, екстракцију карактера и чување у базу. Кодови су поједностављени и избачени су не толико битни делови.

### 6.1. Учитавање докумената

Код у наставку представља део унутар функције која се користи за отварање дијалога и учитавање једне или више слика у апликацију.

```
originalImages = new List<Image>();
// inicijalizuje se dijalog za pretragu fajlova
OpenFileDialog dialog = new OpenFileDialog();
dialog.Multiselect = true;
dialog.Filter = "Image Files(*.BMP; *.JPG)| *.BMP; *.JPG;
//otvara se dijalog za pronalaženje slike
if (dialog.ShowDialog() == DialogResult.OK)
{
    Bitmap bmp;
    //prolazi se kroz odabrane fajlove i čuvaju se slike
    foreach (String imageLocation in dialog.FileNames)
    {
        originalImages.Add((Image)new Bitmap(imageLocation));
    }
    //prva slika se prikazuje u komponenti unutar aplikacije
    Image firstImage = originalImages.First();
    documentShow.Width = firstImage.Width;
    documentShow.Height = firstImage.Height;
    documentShow.Image = firstImage;
}
```

Листинг 6. Учитавање слика

### 6.2. Бинаризација листе слика

Након учитавања слике, да би екстракција имала што боље резултате, препорука је да се изврше филтери за бинаризацију. Имплементација у листингу 7. представља део за бинаризовање слика, а параметри су вредност

контроле *trackBarThreshold* која представља вредност *track-bar* компоненте која је између 0 и 255 која је праг за овај алгоритам, а *cbxThreshold* представља тип алгоритма за бинаризацију који је одабран у тој *check-box* контроли.

```
//inicijalizuje se lista slika
binarizeImages = new List<Image<Gray, byte>>();
//iterira se kroz sve slike
for (int i = 0; i < modifiedImages.Count; i++)
{
    //slika se konvertuje u sivu sliku
    Image<Gray, byte> imgBinarize =
        modifiedImages[i].Convert<Gray, byte>();
    //izvršava se Threshold sa parametrima odabranim u aplikaciji
    CvInvoke.Threshold(imgBinarize, imgBinarize,
        (double)trackBarThreshold.Value, 255,
        (ThresholdType)cbxThreshold.SelectedIndex);
    //dodaju se slike u listu slika
    binarizeImages.Add(imgBinarize);
}

//prikazuje se slika
documentShow.Image = binarizeImages[imageIndex].Bitmap;
```

Листинг 7. Бинаризација листе слика

### 6.3. Имплементација екстракције карактера

У листингу 8. имплементиран позив паралелне функције за екстракцију, а у листингу 9. имплементирани су обраде од позива функција *Tesseract* библиотеке, итерирања кроз препознате резултате обраде, до одсецања карактера са слике и чувања слике у креирани фолдер.

```
//funkcija za pokretanje paralelnog procesa ekstrakcije karaktera
private async void charExtraction_Click(object sender, EventArgs e)
{
    Task taskP = new Task(() => CharExtractionTesseract());
    //započinje se paralelni proces ekstrakcije karaktera
    taskP.Start();
}
```

Листинг 8. Покретање паралелног процеса екстракције

```
//najbitniji delovi funkcije za ekstrakciju karaktera
private async Task CharacterExtractionTesseract() {
    using (var ocr = new TesseractEngine("./tessdata", "eng",
        engineMode))
```

```

//pokreće se Tesseract algoritam kroz dokument bitmapImage
using (var res = ocr.Process(bitmapImage, PageSegMode.Auto))
// Dobija iterator za prolazak kroz OCR rezultat (po simbolima)
using (var iter = res.GetIterator())
// Petlja prolazi kroz sve simbole prepoznate na slici
while (iter.Next(PageIteratorLevel.Symbol))
// Pokušava da dobije okvir oko trenutnog simbola
if (iter.TryGetBoundingBox(PageIteratorLevel.Symbol, out Rect
symbolBounds))
{
// Uzima prepoznati karakter iz trenutnog simbola
char recognizedChar=iter.GetText(PageIteratorLevel.Symbol)[0];
// Dobija poverenje (confidence) za prepoznati simbol
float confidence = iter.GetConfidence(PageIteratorLevel.Symbol);
// Proverava da li je u opsegu validnih ASCII karaktera
if (recognizedChar >= 33 && recognizedChar <= 126) {
    string folderPath =
        $"C:\\CharacterExtraction\\{recognizedChar}";
    // Kreira folder za prepoznati karakter ako već ne
    //postoji
    Directory.CreateDirectory(folderPath);
    // Kreira putanju za čuvanje slike prepoznatog //simbola
    string imagePath = Path.Combine(folderPath,
        $"{Directory.GetFiles(folderPath).Length}.bmp");
    // Iscrtava pravougaonik oko prepoznatog simbola i
    //kreira isecak slike
    using (Image croppedImage = cropImage(bitmapImage, new
    Rectangle(symbolBounds.X1, symbolBounds.Y1,
    symbolBounds.Width, symbolBounds.Height))) {
        // Čuva isecak slike u kreirani folder
        croppedImage.Save(imagePath);
    }
}
}
}
}

```

Листинг 9. Функција за екстракцију карактера

## 6.4. Компресовање и чување карактера у базу

Компресовање у задати оквир већ препознатог карактера и његово чување у нормализованој бази у бинарном формату приказано је у листингу 10. Променљива *normalizedDatabase* представља компресовану базу у бинарном формату. Једни од битних параметара ове базе су *columnNum* и *rowNum* који представљају величину карактера у коју ће бити компресован односно број пиксела у реду и колони, односно висини и ширини.

```

Bitmap img = (Bitmap)Image.FromStream(stream);
// računa se faktor skaliranja
double resize = (img.Width * (int)rowNum) / img.Height <= (int)columnNum
? (double)img.Height / rowNum
: (double)img.Width / columnNum;
// Računa se novua širina (w) i visina (h) slike nakon skaliranja
int w = (int)(img.Width / resize);
int h = (int)(img.Height / resize);
// Kreira se bitmapa sa dimenzijama columnNum x rowNum
Bitmap bmp = new Bitmap((int)columnNum, (int)rowNum);
// Kreira se Graphics objekat da bi se radilo sa bitmapom
using (Graphics gr = Graphics.FromImage(bmp))
{
    gr.FillRectangle(Brushes.White, 0, 0, bmp.Width, bmp.Height);
    // Crta se binarizovana slika na novoj Bitmapi
    gr.DrawImage(BinarizeBitmap(new Bitmap(img, new Size(w, h)), 127), 0,
        (int)rowNum - h, w, h);
}

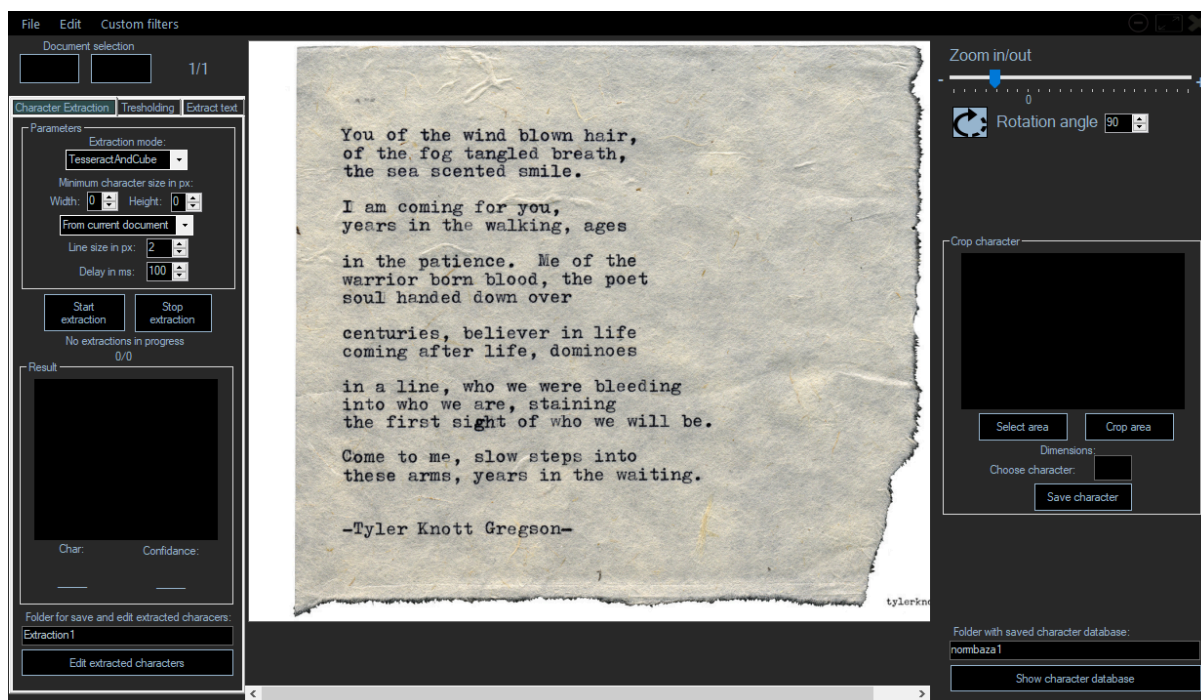
// Dobija se ASCII vrednost trenutnog karaktera (iz imena fajla)
int asciiNum = int.Parse(subdirectoryEntries[i]);
// Izračunaj početni indeks u bazi podataka za ovaj karakter
//(svaki karakter ima svoj prostor u bazi)
int index = (asciiNum - 33) * (int)(rowNum * columnNum);
// Proverava se da li karakter već postoji u bazi podataka
if (saveCharIfNotExist())
{
    // Ažuriraj bazu podataka i sačuvaj binarizovane podatke
    for (int k = 0; k < rowNum; k++)
        for (int j = 0; j < columnNum; j++, index++)
            normalizedDatabase[index] =
                bmp.GetPixel(j, k).R == byte.MaxValue
                ? byte.MinValue
                : byte.MaxValue;
}
// Definiše se putanja gde će se sačuvati slika
string savePath = @"C:\\CharacterExtraction\\" + tbxFileSave.Text + "\\" +
subdirectoryEntries[i] + ".bmp";
// Proverava se da li fajl već postoji, i ako postoji briše se
File.Delete(savePath);
// Čuva se slika na zadatoj putanji
img.Save(savePath);

```

Листинг 10. Промена величине и чување карактера

## 7. Илустрација имплементације

На слици 21. се налази главна форма која се састоји од великог броја делова и контрола. Из главне форме се стиже до свих осталих форми. У наставку су редом описане све могућности апликације, идући редом од почетних задатака и захтева апликације, до завршних обрада.

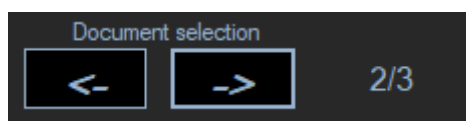


Слика 21. Изглед почетне форме апликације са учитаном сликом

### 7.1. Учитавање скенираног документа

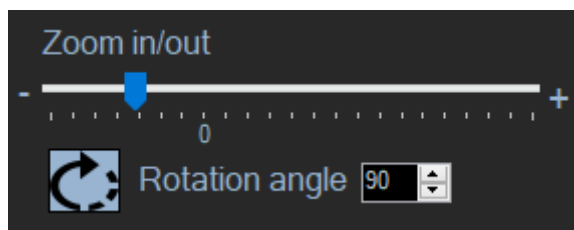
Учитавање слике у *.jpg* или *.bmp* формату се обавља у *menu-strip* контроли одабиром опције *File -> Get images*. Након тога ће се отворити дијалог који омогућава одабир и учитавање једне или више слика.

Када су једна или више слика учитане, у главној форми, у централној контроли апликације, приказује се прва учитана слика. Остале слике могу се листати помоћу дугмића са стрелицама који се налазе у горњем левом углу (слика 22). Прва бројка испод ових дугмића представља тренутно приказан документ, а друга бројка представља укупан број учитаних докумената.



Слика 22. Избор докумената

У горњем десном углу налази се *Track-bar* контрола која служи за зумирање и одзумирање тренутно приказаног документа (слика 23). Овом контролом је могуће управљати и помоћу скрол тастера на мишу након клика на приказану слику. Испод ове контроле налази се дугме са иконом у облику стрелице, које служи за ротацију слике под одређеним углом. Десно од овог дугмета налази се нумеричка контрола која приказује угао ротације.

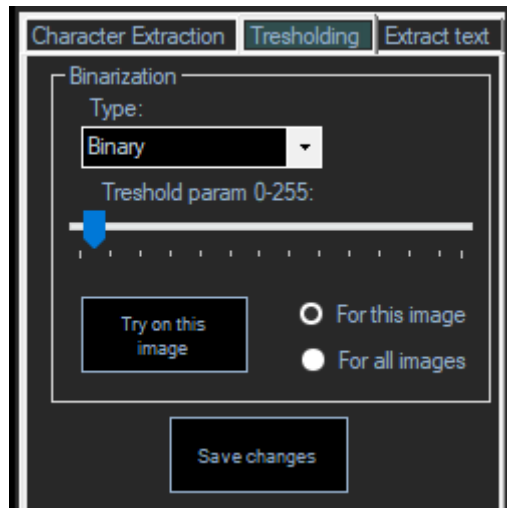


Слика 23. *Track-bar* контрола

## 7.2. Филтери и бинаризација

Део унутар другог таба у таб контроли који се налази лево од приказане централне слике, служи за бинаризацију/*thresholding* слике, приказан је на слици 24. Падајући мени представља главни параметар који одређује на који начин ће се извршити *threshold* слике. Најважнији избори у падајућем менију би били *Binary* и *BinaryInv*, осим овог параметра потребно је изабрати и праг за *thresholding* који се креће од 0-255, померањем ручке на *Track-bar* контроли мењаће се овај параметар, а самим тим ће се и примењивати овај филтер на слици са промењеним параметром. *Binary* опција је најбоље да се користи када су тамна слова на светлој позадини, а у случају да су слова светлија од позадине пожељно је искористити *BinaryInv* опцију, да бисмо добили бинаризовану слику са црним (0) словима и белом (255) позадином.

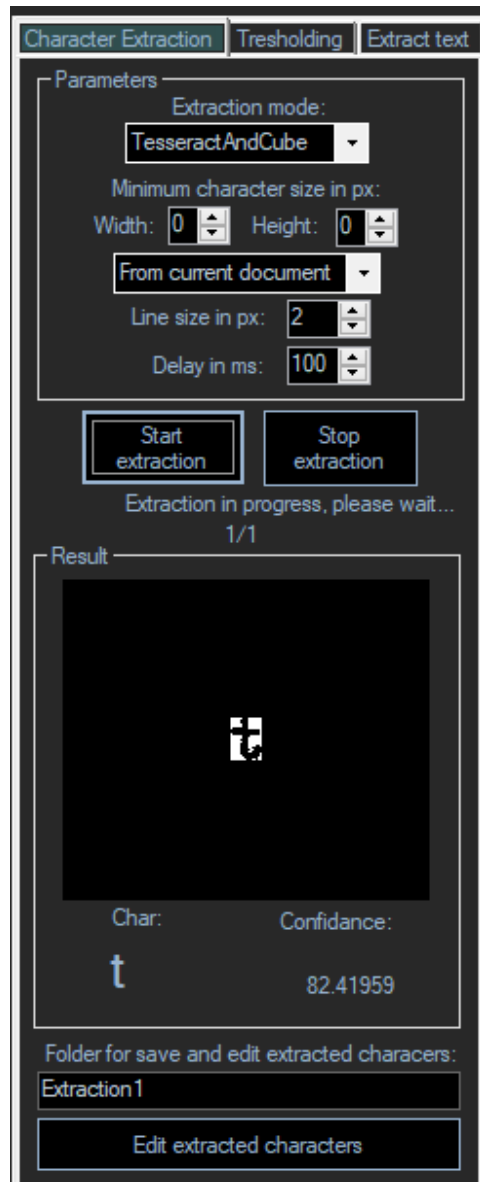
Најбоље је пре свега испробати овај филтер. Дугме са натписом “*Try on this image*” се користи да се испроба филтер и да се бинаризована слика прикаже, слика се тада неће сачувати у апликацији, корисник има могућност да више пута испроба филтер са различитим прагом.



Слика 24. *Thresholding* и бинаризација

Када се корисник одлучи за параметре који му одговарају, потребно је да кликне на дугме са ознаком „*Save changes*“ да би се бинаризована слика трајно сачувала у програму. Избором радио дугмића корисник ће одлучити да ли жели да уради бинаризацију са истим параметрима на све слике ако одабере „*For all images*“ или само за тренутно изабрану слику, ако изабере опцију „*For this image*“.



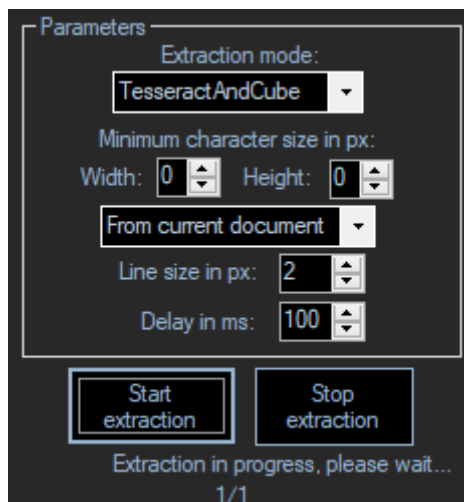


Слика 25. Tab – Сепарација карактера

### 7.3. Сепарација карактера

На слици 25. представљен је цео одељак који служи за сепарацију карактера, на слици 26. је издвојен део за параметре у коме се врши одабир параметара за сепарацију и екстракцију карактера. Најпре је потребно одабрати *Extraction mod* у падајућем менију, *TesseractAndCube mod* је до сада давао најбоље резултате, па је зато постављено као подразумевани мод. Након тога могуће је одабрати колика би била минимална величина карактера који се екстрахује у пикселима, могуће је унети минималну ширину и висину. Други падајући мени представља избор који се односи на то да ли корисник жели да обради све документе редом или жели да обради само документ који је

тренутно приказан. Ако изабере опцију „From current document“, обрадиће само тај, а опција „From all documents“ обрадиће све документе редом.

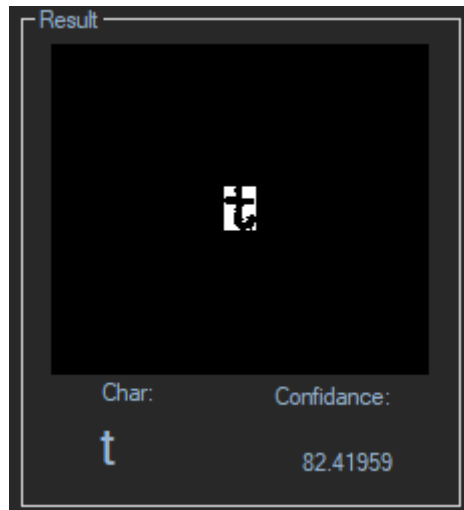


Слика 26. Параметри за екстракцију и сепарацију карактера

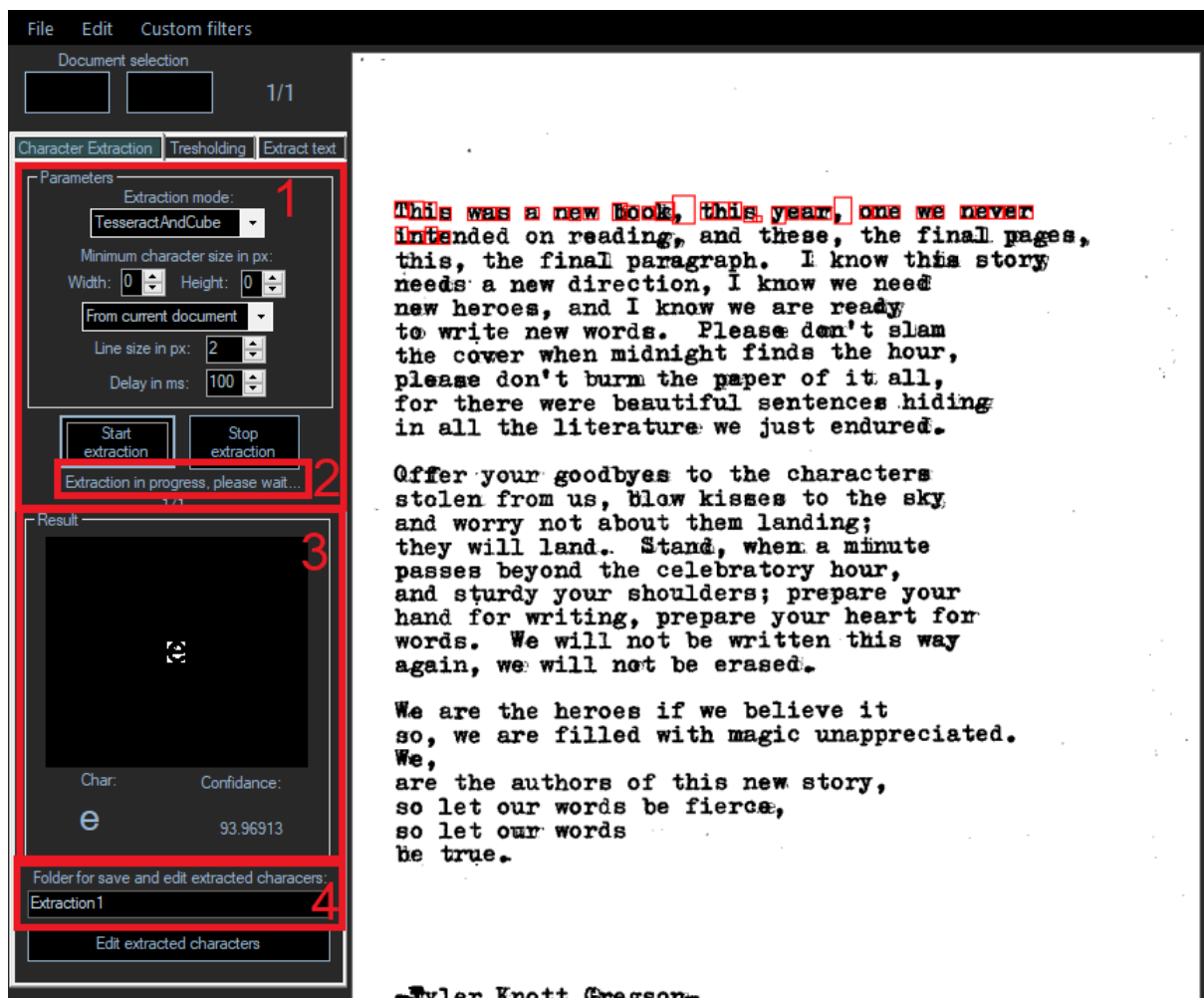
Наредне две бројке представљају величину црвене линије у пикселима која ће уоквиривати карактер по карактер, оним редом којим их обрађује апликација и кашњење у милисекундама које постоји да би корисник визуелно могао да прати обраду док се карактери појављују у контроли на слици 27.

У наставку постоје два дугмета са ознакама „Start extraction“ и „Stop extraction“. Кликком на прво покренућемо процес издвајања и чувања карактера у *file*. Потребно је сачекати од неколико секунди до неколико минута, у зависности од документа и рачунара, док се документ обради и док сепарација карактера крене. У току овог процеса могуће га је зауставити кликом на друго дугме. Испод дугмади постоји ознака која показује напредак процеса и да ли је процес готов, а ознака испод ње означава најпре број документа који се тренутно обрађује и укупан број учитаних докумената.

Слика 27. представља део у коме се приказују карактери редом који су издвојени и који ће се сачувати. Испод ознаке „Char:“ налази се карактер који је препознат, а испод ознаке „Confidence:“ је процена колико је програм сигуран да је то баш тај карактер на слици. Фајл у коме се чувају карактери именован је испод свега овога, на слици 25. и његов назив је „Extraction1“.



Слика 27. Приказ издвојеног карактера и *confidence*



Слика 28. Програм у току екстракције

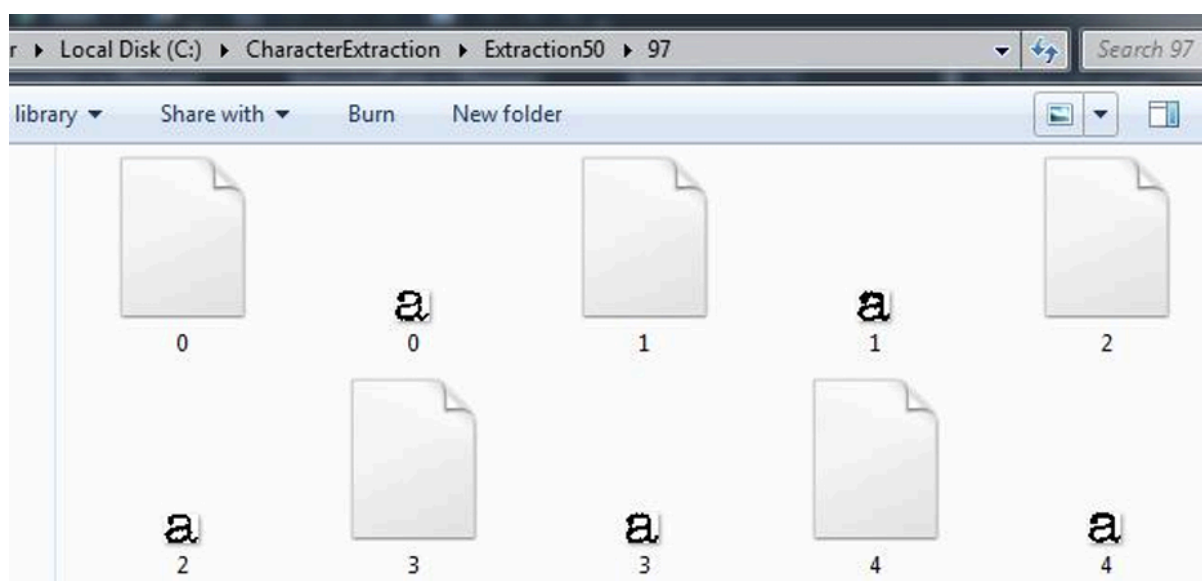
На слици 28. приказан је програм у току екстракције карактера. Црвеним оквирима обележени су региони од интереса везани за процес екстракције:

1. Избор параметара
2. Ознака везана за обраду
3. Део за приказ извучених карактера
4. Фајл у коме се чувају карактери

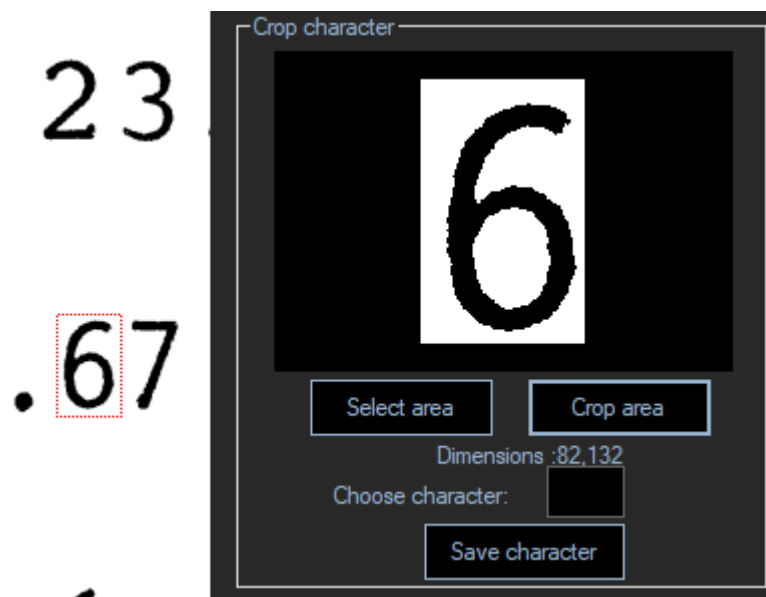
Карактери на слици су такође обележени црвеним оквирима. Програм обележава редом карактер по карактер који се тренутно обрађује и приказује га у делу под бројем 3. На овој слици је сачуван снимак екрана у току обраде, па нису сви карактери још обележени у том тренутку.

## 7.4. Формирање базе карактера

Након екстракције, у одабраном *folder*-у се креира посебан *folder* који добија име по *ASCII* вредности карактера који се чува. У том *folder*-у се чува битмапа извученог карактера и посебан текстуални *file* који има исти назив као битмапа, али без екстензије. Овај текстуални *file* садржи бројку која представља сигурност да је тај карактер одговарајући. На слици 29. може се видети како изгледа један *folder* у коме се чува одређени карактер.



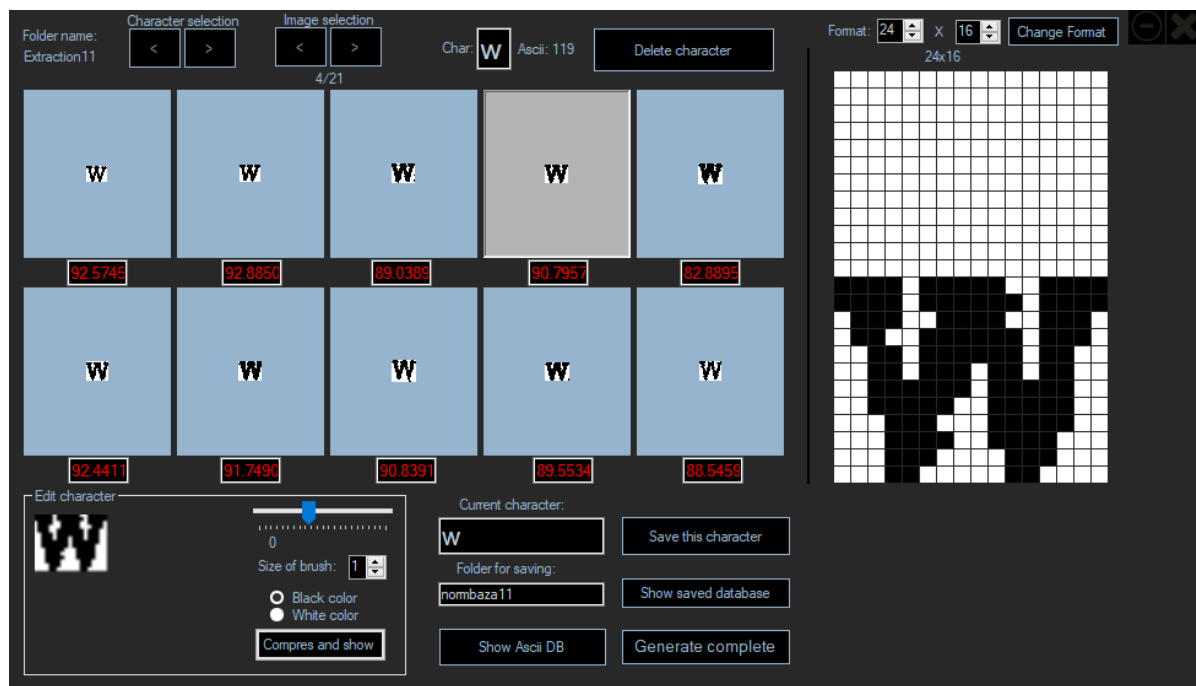
Слика 29. Изглед фолдера (97 *ASCII* је 'а' карактер)



Слика 30. Ручно одсецање карактера

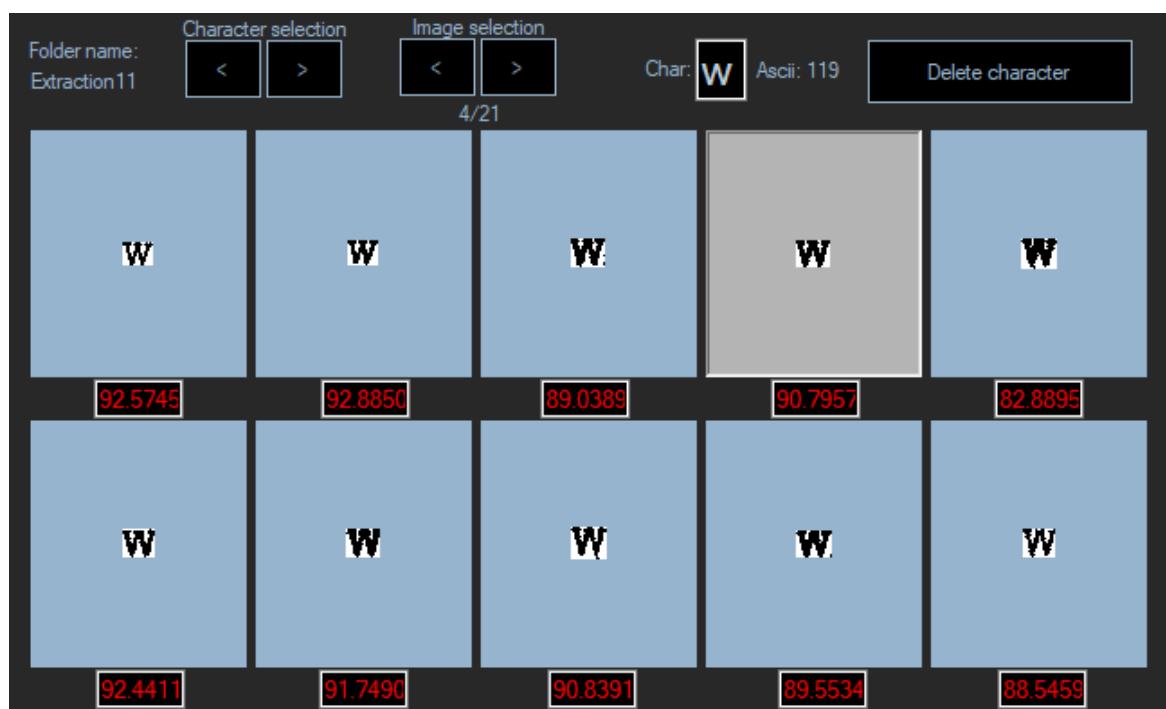
Одељак у десном делу главне форме, као што је приказано на слици 30., служи да се ручно издвоји карактер са слике и дода у базу. Ако желимо да издвојимо карактер, потребно је кликнути на дугме са ознаком „*Select area*“, затим одабрати део на слици који желимо да издвојимо. Део слике ће се обележити испрекиданом црвеном линијом, као на слици 30. у левом делу. Након тога потребно је кликнути на дугме „*Crop area*“. И издвојени део ће се приказати као на слици 30. Затим је потребно додати који карактер је издвојен и кликнути на дугме „*Save character*“. Тиме ће бити сачуван у истом *folder*-у као и карактери који су аутоматски додати.

## 7.5. Уређивање базе карактера



Слика 31. Форма за обраду карактера и нормализовање базе

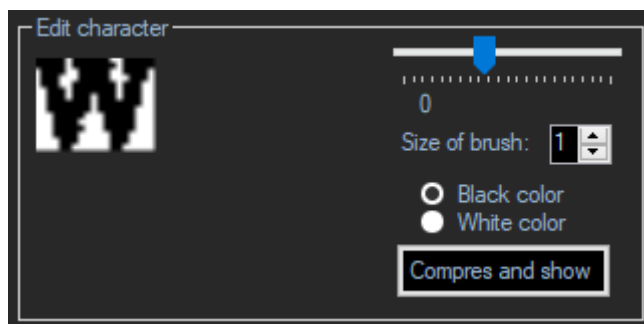
На слици 31. представљена је друга најважнија форма у којој ће се аутоматски или ручним методама одабрати који ће карактер бити представник у бази карактера.



Слика 32. Део за приказ, селекцију и брисање карактера

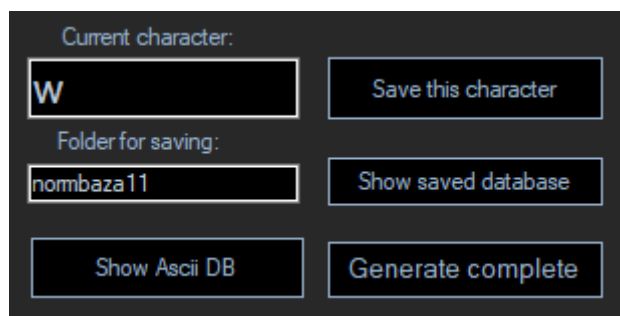
Слика 32. представља део у коме се извучени карактери могу листати, као и брисати. Дугмићи испод ознаке „*Character selection*“ служе за листање различитих *folder*-а, где се један *folder* односи на један карактер. Дугмићи испод ознаке „*Image selection*“ служе за листање извучених карактера из једног *folder*-а, а бројеви испод тих дугмића представљају број тренутно селектованог карактера и укупан број карактера у том *folder*-у. Изабрани карактер је могуће обрисати кликом на дугме „*Delete character*“. Сваки карактер испод слике садржи одговарајући скор, односно број. Што је скор већи, то је већа вероватноћа да је то представљени карактер и да је боља репрезентација тог карактера.

У тренутку када се карактер очита, слика тог карактера се аутоматски пребацује у део за едитовање на слици 33. и аутоматски се компресује у слику величине 24x16 пиксела (или неке кориснички одабране величине) и исцртава се у делу који је приказан на слици 35. Могуће је едитовати изабрани карактер, најпре подесити зум помоћу *track-bar*-а или точкића миша, затим одабрати боју и величину четкице у пикселима и цртати по контроли у којој се налази карактер (слика 33). На крају је могуће поново компресовати карактер кликом на дугме „*Compress and show*“, и компресовани карактер ће бити приказан у делу на слици 38.

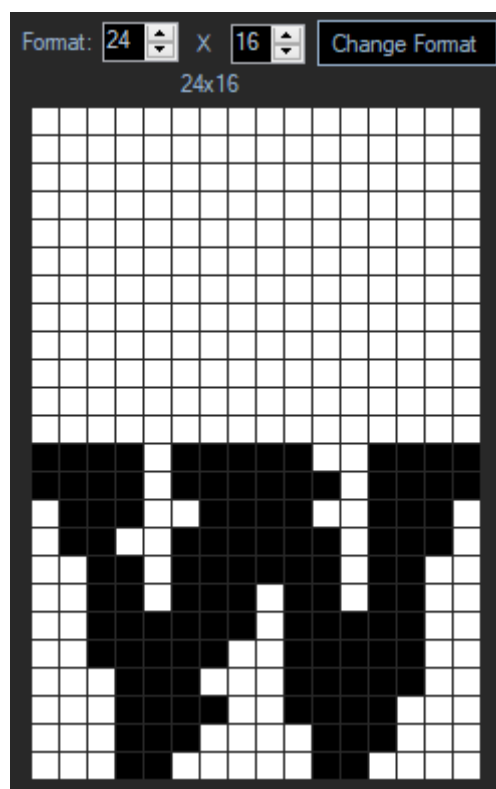


Слика 33. Едитовање бинаризованог карактера

Када корисник измени карактер и компресује га поново у слику одређене величине  $M \times N$ , потребно је да унесе који је то карактер у поље испод ознаке „*Current character*“ и кликне на дугме „*Save this character*“. Карактер ће бити снимљен у нормализовану базу.



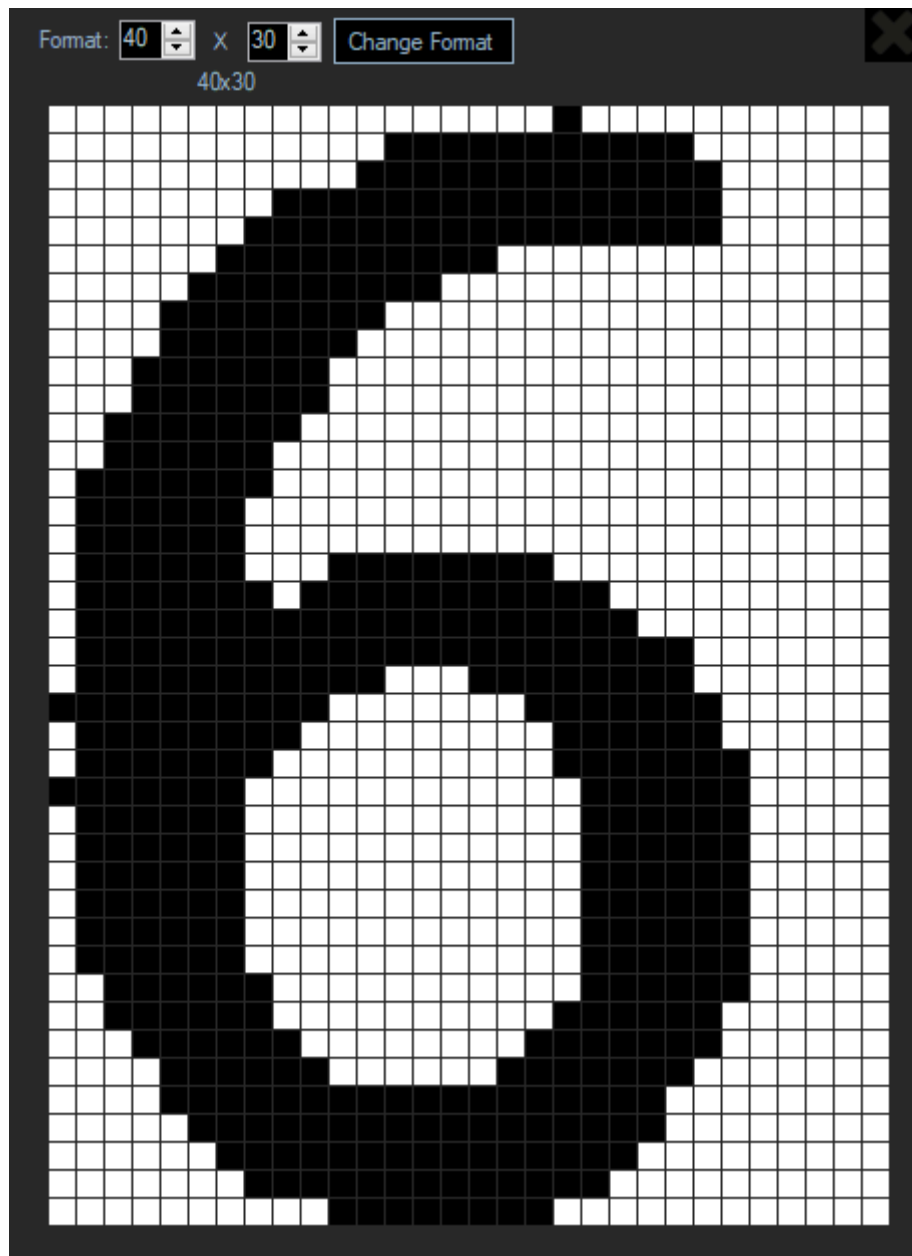
Слика 34. Снимање у базу



Слика 35. Компресован бинаризовани карактер 24x16 пиксела

Део на слици 35. где се налази компресовани карактер могуће је едитовати кликом на квадратиће беле и црне боје, који ће заменити боју у инверзну.





Слика 36. Компресован бинаризовани карактер 40x30 пиксела

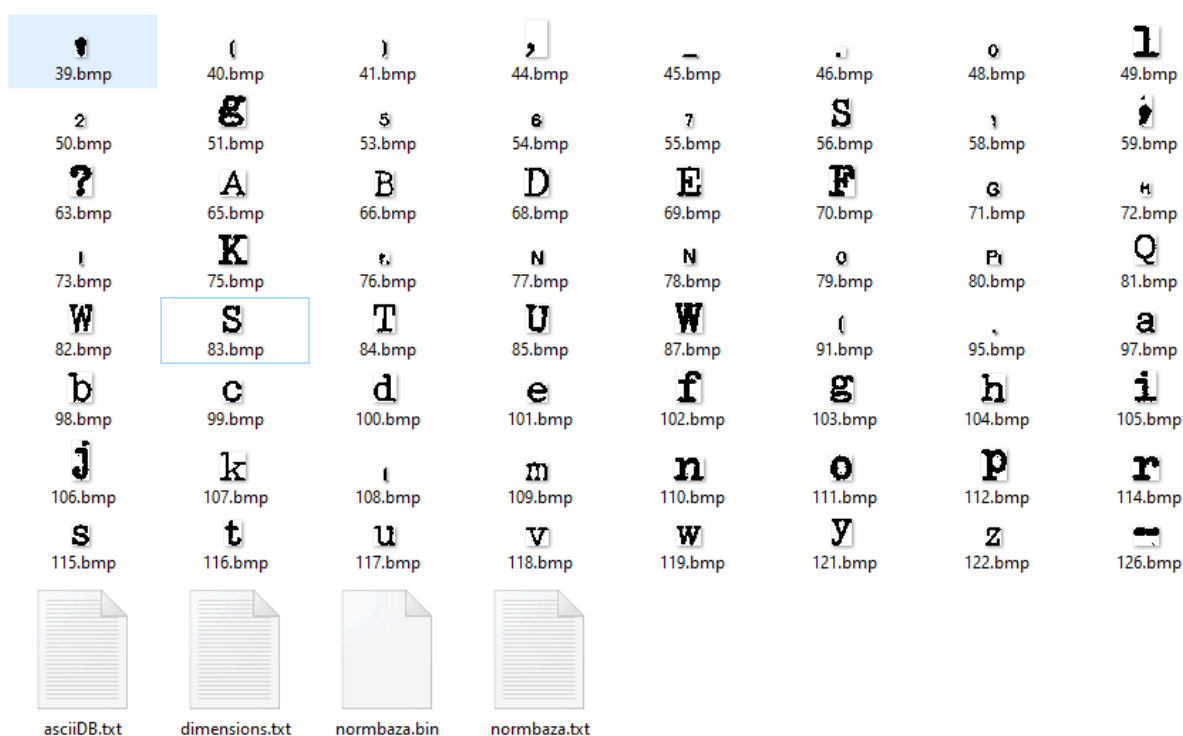
Слика 36. представља компресовани бинаризовани карактер 40x30 пиксела, чија је стандардна величина из апликације промењена. У заглављу слике 36. могу се приметити две контроле за формат, које представљају редове и колоне, и контрола са дугметом „*Change Format*“. У апликацији постоји могућност да се промени формат карактера који се чува у бази. Потребно је изменити редове и колоне у овим контролама, а кликом на дугме наредни карактери ће се компресовати у нову величину. Стара база ће бити обрисана јер није могуће чувати карактере различитих димензија.

## 7.6. Нормализација

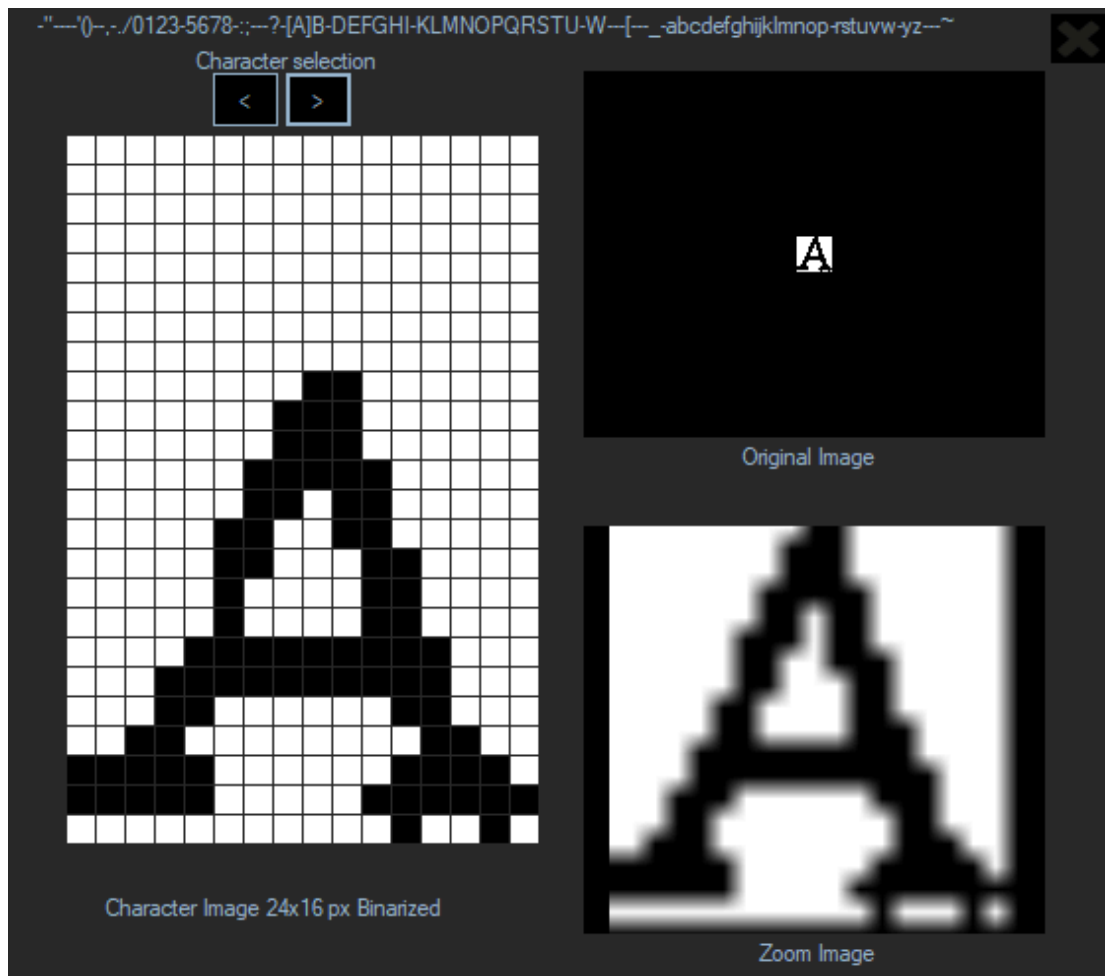
Да би се едитовани карактер снимиио у базу, потребно је одабрати име базе тако што ће се у делу на слици 34. испод ознаке „*Folder for saving:*“ унети име тог фолдера за снимање.

Поред ручног додавања карактера, могуће је генерисати комплетну нормализовану базу. За генерисање комплетне нормализоване базе карактера користи се дугме „*Generate complete*“ са слике 34. Ова опција ће изабрати карактере са најбољим скором и додати их у нормализовану базу. Након избора опције, појавиће се пар порука (*MessageBox*) са питањима везаним за креирање нове базе, попут тога да ли да остану сачувани претходни карактери.

Нормализована база се чува у облику пар различитих формата, поред фајлова *normbaza.bin*, *normbaza.txt*, *asciiDB.txt* и *dimensions.txt*, чува се и слика карактера у *BMP* формату која има назив као *ASCII* вредност тог карактера (Слика 37).



Слика 37. Изглед фолдера нормализоване базе



Слика 38. Форма за приказ нормализоване базе карактера

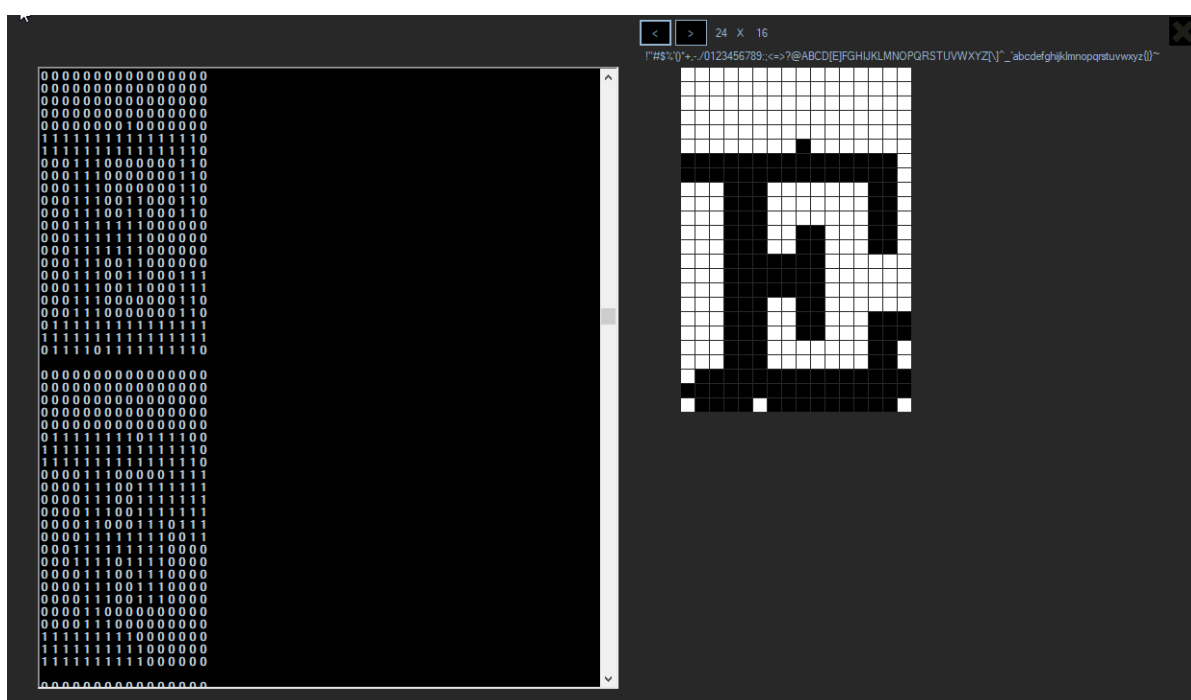
Приказ нормализоване базе на слици 36. у средини десно налази се део преко ког се приступа нормализованој бази ради прегледа сачуваних карактера. Контрола односно дугме „*Show saved database*“ омогућава отварање нове форме за преглед сачуваних карактера, приказане на слици 38.

Испод ознаке „*Character selection*“ налазе се дугмићи који се користе за листање карактера из базе. Са десне стране приказан је компресовани карактер, а лево су приказане оригинална слика и зумирана слика.

Сви фолдери и фајлови који се користе као подаци у програму чувају се у фолдеру „*C:\CharacterExtraction*“, који ће бити аутоматски креиран од стране апликације.

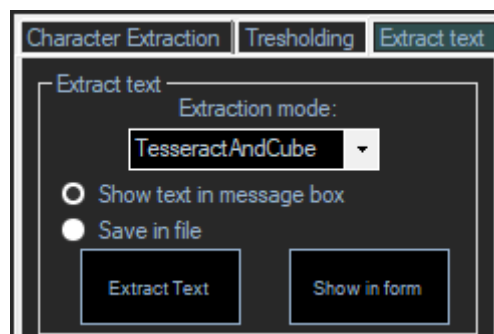
## 7.7. Приказ формата у ASCII редоследу

Поред свих до сада описаних контрола са слике 34. постоји и дугме „*Show Ascii DB*“, које ће најпре отворити дијалог за претрагу текстуалних фајлова у фолдерима сачуваних база у „*C:\CharacterExtraction*“, где је могуће изабрати неки од текстуалних фајлова посебног *ASCII* формата. Могуће је отворити и фајл овог формата из било ког дела фајл система на рачунару. Ако је фајл у исправном формату, отвориће се форма са слике 39. Лево ће бити приказан део сировог фајла овог формата, а десно су приказани карактери који се листају помоћу контрола у горњем делу, који су извучени из овог формата. Овај формат ће такође бити запамћен у истој величини *MxN* као и бинарни компресовани формат, а у прва два реда овог текстуалног фајла исписани су број редова и број колона.



Слика 39. Форма за приказ формата *ASCII* типа

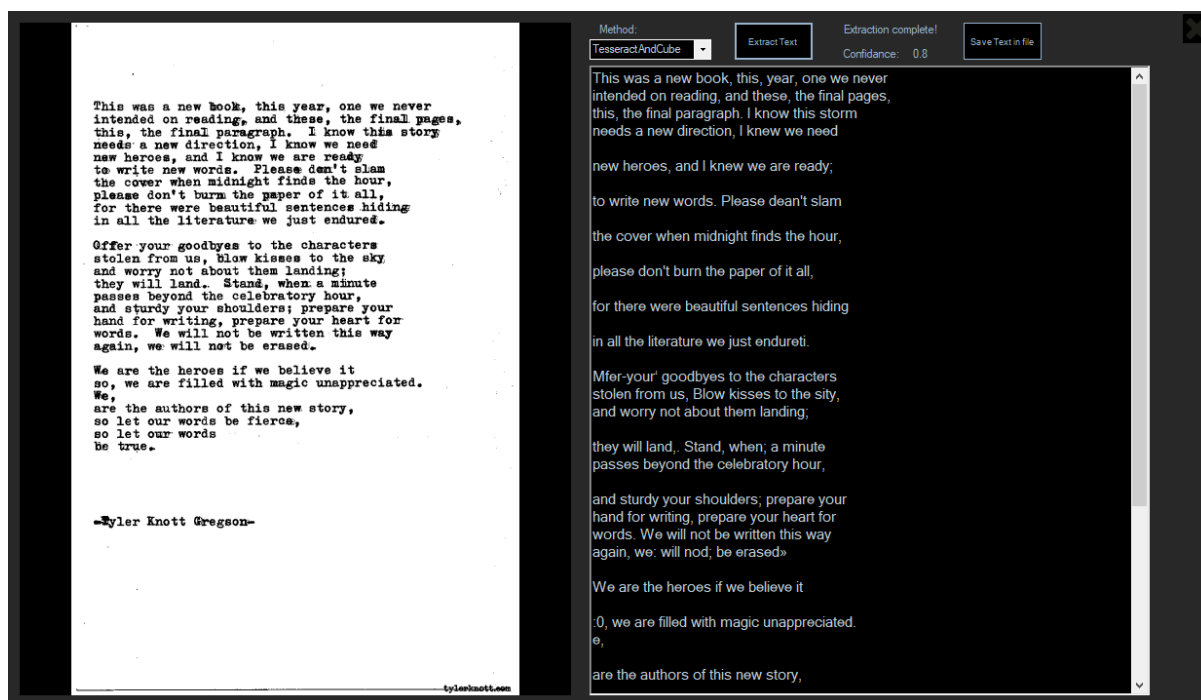
## 7.8. Додатни део апликације



Слика 40. Одељак за екстракцију комплетног текста са документа

Пре него што се пређе на остале ствари везане за крајње обраде добијених карактера из текста, биће описан последњи део на главној форми. Део са слике 40. служи за добијање комплетног текста са документа. Могуће је најпре одабрати мод за екстракцију, а затим један од два понуђена радио дугмета. Може се одабрати да се текст из тренутно изабраног документа прикаже у *MessageBox* форми или да се одмах сними директно у фајл. Након избора потребно је притиснути дугме „*Extract Text*“. Пре креирања фајла отвориће се дијалог у коме ће се изабрати место, име и екстензија фајла у који желимо да снимимо текст.

Ако желимо бољи преглед, кликом на дугме „*Show in form*“ отвориће се нова форма у којој ће бити приказани изабрани текстуални документ и поље у коме ће се исписати екстраховани текст. Потребно је поново изабрати метод и потврдити. Након добијања текста, постоји могућност да се текст ручно едитује како би се исправиле евентуалне грешке програма. Када се заврши са додатним едитовањем, могуће је снимити текст избором дугмета „*Save Text in file*“. Поново ће се појавити дијалог у коме ћемо изабрати локацију и назив фајла у који ће се уписати текст (Слика 41).



Слика 41. Упоредни приказ документа и екстрахованог текста

## 8. Закључак

Развој дигиталне обраде слике и *OCR* технологије омогућио је значајне напретке у многим областима. Кроз овај рад, истражене су могућности и изазови развоја такве апликације, са циљем да се допринесе даљем напретку у области дигиталне обраде слике и *OCR* технологије.

Ова апликација комбинује напредне технике обраде слике и *OCR* препознавања како би омогућила корисницима да једноставно екстрактују текст из слика и креирају јединствене фонтове. Ова технологија има широк спектар примена, укључујући дигитализацију старих рукописа, персонализацију дизајна, те олакшавање рада у креативним индустријама.

У закључку, развој апликације не само да демонстрира практичну примену напредних технологија обраде слике и *OCR*-а, већ и отвара врата за нове иновације и могућности у дигиталном свету. Иако постоје технички изазови, као што су прецизност препознавања карактера и оптимизација перформанси, континуирани напредак у овој области обећава још веће потенцијале за будућност. Интеграцијом ових технологија у свакодневне алате, олакшава се процес креирања дигиталних садржаја, омогућавајући корисницима да брзо и ефикасно остварују своје идеје.

Даље истраживање и развој у овом домену ће несумњиво донети нове методе и побољшања, чиме ће се проширити могућности и унапредити квалитета апликација које се ослањају на дигиталну обраду слике и *OCR* технологију. Резултати апликације овог рада могу наћи велику примену тиме што могу бити већ спреман скуп података за неке даље обраде алгоритама Машинског учења. Довољно је обрадити само пар страница да се добијемо базу карактера односно фонт једне целе књиге. Овај рад представља значајан корак у том правцу, истовремено служећи као инспирација за будуће пројекте и иновације.

## 9. Литература

- [1] K. R. Dongur, P. Tandekar, and S. K. Purve, "Digital Image Processing: Its History and Application," \*International Journal of Engineering Research and Technology\*, vol. 6, June 2022.
- [2] R. W. Smith, "The Extraction and Recognition of Text from Multimedia Document Images," Ph.D. dissertation, Dept. Eng., Univ. of Bristol, Bristol, U.K., 1987.
- [3] I. Marosi, "Industrial OCR approaches: architecture, algorithms and adaptation techniques," in \*Document Recognition and Retrieval XIV\*, SPIE, Jan. 2007.
- [4] S. Impedovo, Ed., \*Fundamentals in Handwriting Recognition\*. Berlin, Germany: Springer-Verlag, 2012.
- [5] S. V. Rice, G. Nagy, and T. A. Nartker, \*Optical Character Recognition: An Illustrated Guide to the Frontier\*. Berlin, Germany: Springer, 2012.
- [6] R. C. Gonzalez and R. E. Woods, \*Digital Image Processing\*, 4th ed. London, U.K.: Pearson, 2018.
- [7] S. V. Rice, F. R. Jenkins, and T. A. Nartker, "The Fourth Annual Test of OCR Accuracy," \*International Journal of Document Analysis and Recognition\*, May 2013.



## 10. Списак слика и листинга

Слике:

Слика 1. Креирање хистограма на основу сиве слике.....	9
Слика 2. Слика руже и хистограм пре и после филтера бинаризације.....	10
Слика 3. Пример изједначавања хистограма.....	11
Слика 4. Примери за (a) <i>Nearest-neighbor</i> , (b) <i>Bilinear</i> , (c) <i>Bicubic</i> , (d) Оригинал.....	12
Слика 5. Скица патента Густава Таушека.....	13
Слика 6. OCR фонт.....	16
Слика 7. Откривање карактеристика карактера.....	17
Слика 8. Пример обележеног закривљеног текста.....	20
Слика 9. Пример детекције фиксне ширине слова.....	20
Слика 10. Текст са променљивим размаком између речи.....	20
Слика 11. Могуће тачке за одсецање.....	21
Слика 12. Изломљени карактери.....	21
Слика 13. График са поређењем различитих OCR алата.....	22
Слика 14. Дијаграм тока.....	27
Слика 15. Дијаграм случајева коришћења.....	28
Слика 16. Секвенцијални дијаграм примене филтера над сликама.....	29
Слика 17. Секвенцијални дијаграм екстракције карактера.....	30
Слика 18. Дијаграм <i>Windows Form</i> компоненти.....	31
Слика 19. Дијаграм активности паралелних радњи апликације.....	32
Слика 20. Дијаграм протока података.....	33
Слика 21. Изглед почетне форме апликације са учитаном сликом.....	38
Слика 22. Избор докумената.....	38
Слика 23. <i>Track-bar</i> контрола.....	39
Слика 24. <i>Thresholding</i> и бинаризација.....	40
Слика 25. <i>Tab</i> – Сепарација карактера.....	41
Слика 26. Параметри за екстракцију и сепарацију карактера.....	42
Слика 27. Приказ издвојеног карактера и <i>confidence</i> .....	43
Слика 28. Програм у току екстракције.....	43
Слика 29. Изглед фолдера (97 ASCII је 'а' карактер).....	44
Слика 30. Ручно одсецање карактера.....	45
Слика 31. Форма за обраду карактера и нормализовање базе.....	46
Слика 32. Део за приказ, селекцију и брисање карактера.....	46
Слика 33. Едитовање бинаризованог карактера.....	47
Слика 34. Снимање у базу.....	48
Слика 35. Компресован бинаризовани карактер 24x16 пиксела.....	48
Слика 36. Компресован бинаризовани карактер 40x30 пиксела.....	49

Слика 37. Изглед фолдера нормализоване базе.....	50
Слика 38. Форма за приказ нормализоване базе карактера.....	51
Слика 39. Форма за приказ формата <i>ASCII</i> типа.....	52
Слика 40. Одељак за екстракцију комплетног текста са документа.....	53
Слика 41. Упоредни приказ документа и екстрахованог текста.....	54

#### Листинзи:

Листинг 1. Глобални <i>thresholding</i> .....	24
Листинг 2. Адаптивни <i>thresholding</i> .....	24
Листинг 3. <i>Otsu</i> метод.....	25
Листинг 4. Верзија <i>.NET</i> Окружења.....	26
Листинг 5. Верзије главних библиотека.....	26
Листинг 6. Учитавање слика.....	34
Листинг 7. Бинаризација листе слика.....	35
Листинг 8. Покретање паралелног процеса екстракције.....	35
Листинг 9. Функција за екстракцију карактера.....	36
Листинг 10. Промена величине и чување карактера.....	37