IZDELAVA MODULA ZA POMOČ PRI VNOSU PODATKOV IN KLASIFIKACIJO SLIK ZA PROGRAMSKO OPREMO GALIS

Diplomsko delo

Študent:	Marko Kokol
Študijski program:	univerzitetni program
	računalništvo in informatika
Smer:	programska oprema
Mentor:	red. prof. dr. Milan Zorman
Somentor:	Sašo Zagoranski, univ. dipl. inž. rač. in inf.

ZAHVALA

Za občasno dobro merjeno (in mišljeno) brco se zahvaljujem mentorju dr. Milan Zormanu, somentorju Sašu Zagoranskemu, svojemu prvemu mentorju dr. Kokolu in lektorici Tanji Kšela.

Posebej bi se želel zahvaliti tudi direktorici Nini Zagoranski za vso pomoč v času nastajanja diplome ter ženi Katji Kokol, staršem, krušnim staršem in dovolj pridnemu otroku.

Nazadnje, a ne najmanj pomembno, gre posebna zahvala tudi vsem dobrim in slabim profesorjem FERI-ja, približno v duhu zakona optimizma: »*Noben poizkus ne pomeni popolnega neuspeha - vedno lahko služi za negativen primer.*« [1]

IZDELAVA MODULA ZA POMOČ PRI VNOSU PODATKOV IN KLASIFIKACIJO SLIK ZA PROGRAMSKO OPREMO GALIS

Ključne besede: inteligentni sistemi, segmentacija slik, razpoznavanje slik, polno besedilno iskanje, prilagojeni predlogi

UDK: 004.81:004.89(043.2)

Povzetek:

V okviru diplomskega dela je bil izdelan nov modul za komercialni programski paket Galis, ki ga razvija podjetje Semantika d.o.o. in ga uporablja večina slovenskih muzejev, prisoten pa je tudi na drugih trgih držav CE regije.

Razvit modul dodaja metode strojnega učenja in umetne inteligence z namenom povečanja produktivnosti muzejskih strokovnjakov, ki produkt uporabljajo pri svojem delu, predvsem z omogočanjem hitrejšega dostopa do informacij, avtomatizacije ponavljajočih se opravil in samodejnega razpoznavanja predmetov na vnesenih slikah.

Izdelane rešitve smo tudi preverili in potrdili, da so smiselna dopolnitev programskega paketa.

DEVELOPMENT OF AN INTELLIGENT MODULE FOR AUTOMATIC IMAGE CLASSIFICATION AND USER ENTRY AUTOMATION FOR GALIS SOFTWARE

Key words: intelligent systems, image segmentation, image classification, full text search, user suggestions

UDK: 004.81:004.89(043.2)

Abstract:

The goal was to develop a new module within the context of an existing commercial content management system for museums (called Galis) that is used by most of museums in Slovenia and is also present on other markets in the CE region.

The developed module adds intelligent capabilities to the Galis software with the purpose of improving productivity of museums' professionals utilizing the software, by improving access to information stored in the system and automating certain tasks.

After being developed the module was validated and tested to confirm the feasibility of including it in the Galis software package.

KAZALO VSEBINE

KAZALO SLIK

KAZALO TABEL

SEZNAM UPORABLJENIH KRATIC

CLR: Common Language Runtime

COM: Component Object Model

DLL: Dynamic Link Library

DNN: Deep Neural Network

ES: Evropska Skupnost

EU: Evropska Unija

GPU: Graphical Processing Unit

IDE: Integrated Development Environment

IJW: It Just Works

JVM: Java Virtual Machine

MS: Microsoft

REST: REpresentational State Transfer

SQL: Structured Query Language

TF-IDF: Term Frequency – Inverse Document Frequency

WS: Web Services

1 UVOD

»And to this end they built themselves a stupendous super-computer which was so amazingly intelligent that even before its data banks had been connected up it had started from I think therefore I am and got as far as deducing the existence of rice pudding and income tax before anyone managed to turn it off.[1]*«* [2]

Da se tehnologija danes razvija neverjetno hitro in je prisotna praktično v vseh segmentih našega ustvarjanja se strinjamo skoraj vsi, zato se bomo v začetku uvoda raje posvetili pomembnosti poznavanja naše preteklosti in kulturne dediščine.

Že Lizbonska pogodba poudarja pomen kulture, njen uvod se namreč sklicuje na zajemanje »navdiha iz kulturne, verske in humanistične dediščine Evrope«, znotraj EU in Slovenije pa potekajo številni programi za ohranitev kulturne dediščine [3].Tudi evropska strategija za kulturo za leto 2015-2018 kot enega ključnih ciljev navaja enostaven dostop do kulturnih virov. [4]

V podjetju Semantika je zato eno naših temeljnih poslanstev pomoč institucijam, ki se ukvarjajo z ohranitvijo kulturne dediščine pri nas in v drugih državah EU, primarno muzejev in arhivov. Naš primarni produkt, ki ga uporablja večina slovenskih muzejev in s katerim smo od leta 2015 prisotni tudi na hrvaškem trgu, je muzejski dokumentacijski sistem Galis. Navedeni sistem je povezan tudi z našimi mobilnimi in spletnimi platformami (npr. http://Museums.EU).

Dokumentiranje zgodovine je ena izmed ključnih nalog muzejev in si je danes skoraj ne moremo predstavljati brez ustrezne informacijske podpore. Ker gre pri tem že v osnovi za precej zamuden proces, v okviru katerega je potrebno popisati velike količine podatkov, je torej zelo smiselno razmišljati o načinih pohitritve takega vnosa – kjer se kot enega možnih načinov takoj domislimo uporabe metod umetne inteligence. Ker moramo pri tem zagotovo poseči tudi na področje obdelave slikovnega gradiva, lahko že pred podrobno analizo ugotovimo, da bo šlo za verjetno časovno precej zahtevne operacije.

---

[1]Prevod: Zato so zgradili nenavadno močan super-računalnik, ki je bil tako neverjetno inteligenten, da je že preden so priklopili njegove spominske celice začel iz »Mislim torej sem« in prišel vse do odkritja riževega narastka in davka iz dohodka, preden so ga uspeli izključiti.

To pa pomeni, da moramo razmišljati o strojni opremi, ki jo bomo uporabljali za analizo podatkov, saj je namen izvedbe pohitritev dela uporabnikov, kar pomeni, da mora računalnik stvari izvesti dovolj hitro, da bo uporabnik celokupno porabil manj časa. Pri tem moramo upoštevati dejstvo, da govorimo o razvoju modula za komercialno programsko opremo, ki se uporablja na realni strojni opremi v muzejih.

V zadnjih letih na področju razvoja strojne opreme sicer bolj kot napredek v osnovni hitrosti procesorjev, opažamo usmeritev v višanje stopnje vzporednega izvajanja (*ang. parallel processing*), poseben napredek pa je viden tudi na področju procesiranja matematičnih problemov na grafičnih procesorskih enotah (*ang graphical processing units – GPU)*. Hkrati so večji proizvajalci grafičnih procesnih enot izdali nekaj standardov, ki omogočajo koriščenje njihovih funkcij v splošno-namenski programski opremi, predvsem OpenCL in CUDA, kar pomeni, da lahko sedaj na »navadnih« namiznih računalnikih realno izvajamo nekatere operacije, ki so bile prej rezervirane za namenske delovne postaje. [5] [6]

V okviru diplomskega dela smo se torej lotili reševanja problema hitrejšega vnašanja podatkov v obstoječo programsko rešitev za muzeje Galis, ker pa je celoten problem preobsežen za obseg tega diplomskega dela, smo se omejili na nekaj ključnih medsebojno povezanih problemov za doseganje cilja:

- identifikacija potencialnih vnosov za polja na podlagi vrednosti drugih polj (npr. samodejno izpolnjevanje avtorja na podlagi izbrane zbirke, ipd.);
- hitro iskanje po podatkih, ki so že v podatkovni bazi;
- segmentacija naloženih slik;
- razpoznavanje naloženih slik.

V uvodu smo zelo na kratko predstavili motivacijo in osnovni problem, ki ga ta diplomska naloga poskuša identificirati in rešiti.

V drugem poglavju na kratko opišemo motivacijo za izdelavo diplomske naloge, probleme, ki jih naloga poskuša rešiti in opišemo trenutno stanje tehnologij in platform ter rešitev na trgu, ki se ukvarjajo ali bistveno posegajo na področje tega diplomskega dela.

V tretjem poglavju se nato osredotočimo na specifične tehnologije, ki smo jih izbrali za izvedbo rešitve, opišemo njihove ključne značilnosti, prednosti in omejitve, ter opišemo, kako smo jih medsebojno povezali. Opišemo tudi zasnovano rešitev, njene glavne zahteve in značilnosti, njeno umestitev v programski sistem Galis in pregledamo ključne algoritme in rešitve, ki smo jih zasnovali v okviru diplomskega dela.

V četrtem poglavju nato preverimo razvito rešitev na nekaj testnih vzorcih, izmerimo njene odzive in preverimo njeno praktično uporabnost na omejenih vzorcih. Pogledamo tudi nekaj statistik razpoznave zanimivih objektov in poskušamo identificirati načine, da bi dobljene rezultate dodatno izboljšali.

Na koncu v petem poglavju povzamemo ugotovitve celotne diplomske naloge, preverimo nekatere potencialne možnosti izboljšave in začrtamo možnosti za nadaljnji razvoj rešitve. Pogledamo tudi, kateri deli rešitve so že zadostni, da jih lahko neposredno umestimo v komercialno aplikacijo in katere bo potrebno še dodatno dodelati.

2 STANJE TEHNOLOGIJ IN TRGA TER MOTIVACIJA

»The factory of the future will have only two employees, a man and a dog. The man will be there to feed the dog. The dog will be there to keep the man from touching the equipment.[2]« - *Warren G. Bennis* [7]

2.1 Motivacija za razvoj modula za pomoč pri vnosu podatkov in razpoznavanja slik

Zaradi velike količine podatkov, ki jo morajo zaposleni v muzejih voditi, je vnos teh podatkov lahko zelo zamuden, hkrati pa smo na Semantiki pri delu z uporabniki na podlagi njihovih povratnih informacij opazili, da jim lahko zelo pomagamo, predvsem če avtomatiziramo nekatera ponavljajoča opravila in omogočamo hitro in učinkovito iskanje po njihovih podatkovnih bazah.

V okviru analize smo tako identificirali nekaj prioritetnih točk, ki smo jih želeli vključiti v nove različice programa, na podlagi te analize pa je nastal tudi seznam zanimivih komponent, ki smo ga že opisali v okviru poglavja 1 in jih na kratko ponovno povzemamo v poglavju 2.2.

Pri tem je potrebno poudariti, da je bil namen razvitih modulov vključitev v obstoječ komercialni produkt, zato smo se pri izbiri upoštevali tudi kompatibilnost z že uporabljenimi tehnologijami v programskem paketu Galis ter stanje komercialne razvitosti rešitev in njihove licenčne omejitve.

2.2 Stanje tehnologij in trga

Kot smo povedali v uvodu, smo se v okviru diplomske naloge lotili reševanja problema hitrejšega vnosa podatkov v muzejski dokumentacijski sistem Galis. V okviru tega poglavja bomo preverili tehnologije, ki smo jih pregledali kot potencialno zanimive. Pogledali bomo torej:

- programsko opremo Galis,
- rešitve indeksacije podatkov in učinkovitega iskanja po njih,

---

[2]Prevod: Pisarna prihodnosti bo imela samo dva zaposlena: človeka in psa. Človek bo tam, da bo hranil psa, pes pa zato, da bo preprečil človeku, da se dotakne opreme. [7]

- rešitve računalniškega vida za segmentacijo in klasifikacijo slik.

2.2.1 Pregled programske opreme Galis

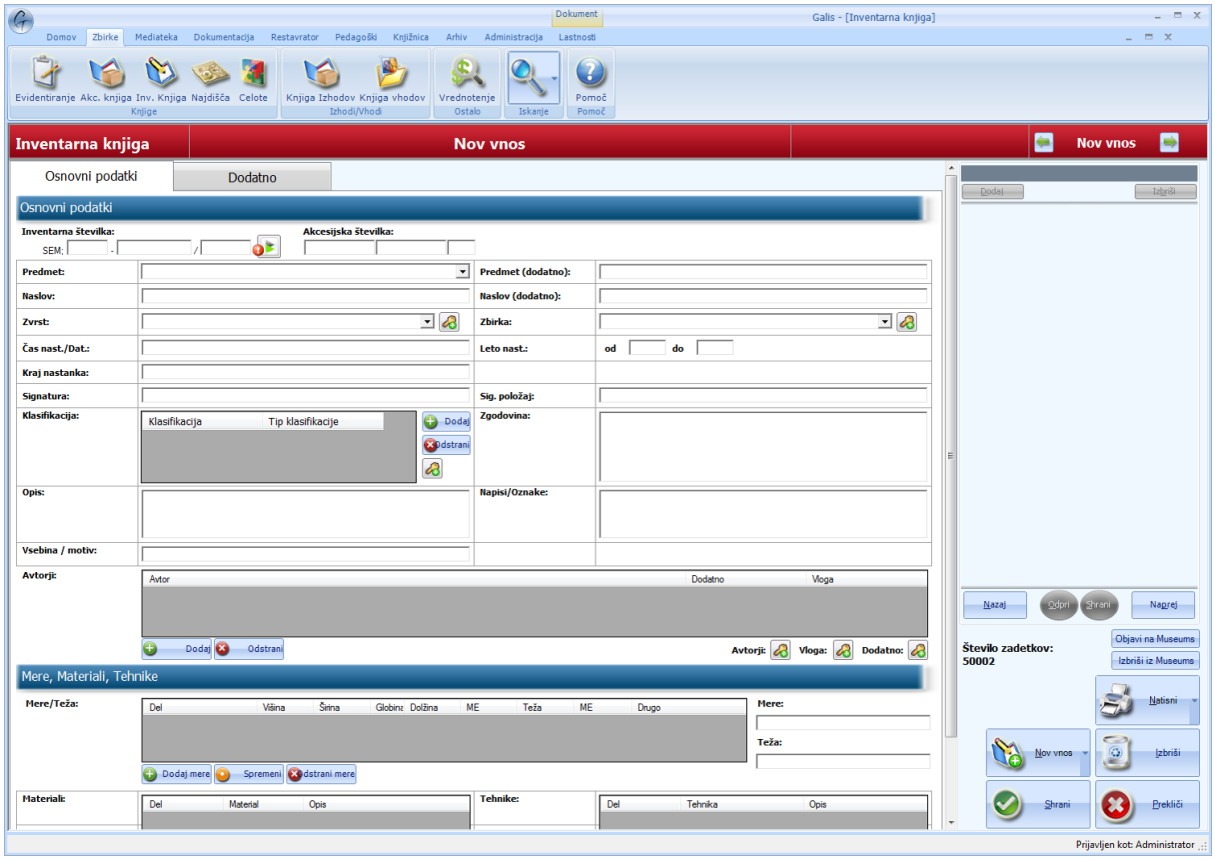
Muzejski informacijski sistem Galis je najbolj razširjen program za vodenje muzejske dokumentacije v Sloveniji. Uporablja ga okrog 70 slovenskih muzejev in galerij, pa tudi že prvi uporabniki v tujini. [8] [9]

Namizno aplikacijo Galis dopolnjuje vrsta povezanih storitev. Te vključujejo:

- sistem za samodejno varnostno kopiranje podatkov v oblak;
- sistem za objavo podatkov na spletu in v mobilnih aplikacijah;
- vtičnik za objavo zbirk na obstoječi spletni strani ustanove;
- sistem za izdelavo spletne strani ustanove.

V razvoju je tudi spletna različica namizne aplikacije.

Galis in vse povezane produkte oz. storitve od leta 2007 razvija, trži in vzdržuje podjetje Semantika.



Slika 2.1: Uporabniški vmesnik programa Galis

Moduli programske opreme Galis

Program Galis je sestavljen iz osmih modulov, ki v celoti pokrivajo poslovanje muzeja in so popolnoma povezani med seboj. Posamezna licenca programa je vezana na uporabnika in vključuje vse module, ki jih na kratko opisujemo v nadaljevanju. [10] [11]

1. **Modul Zbirke**

Modul Zbirke je prvi modul v sistemu in je osnova za delo v kulturni ustanovi. Predstavlja podporo za vodenje elektronske inventarne knjige in omogoča dokumentiranje predmetov, ki so del zbirke muzeja ali galerije.

Sestavljajo ga maske:

- Evidentiranje,
- Akcesijska knjiga,
- Inventarna knjiga,
- Najdišča,
- Celote,
- Knjiga izhodov (Reverzi),
- Knjiga vhodov,
- Vrednotenje.

2. **Modul Restavrator**

Modul Restavrator omogoča natančen popis restavratorskih postopkov:

- Popis osnovnih restavratorskih posegov,
- Beleženje predhodnih posegov,
- Beleženje analiz,
- Popis stanja,
- Popis izvedenih postopkov, materialov in ur.

Modul je mogoče uporabiti za katerokoli vrsto materiala (npr. kovina, tekstil, slike, ...). Restavratorjem omogoča samostojno uporabo (za zunanje posege) ali enostavno povezavo z obstoječimi podatki v Inventarni knjigi, s čimer se uporabniki izognejo dvojnemu delu.

3. **Modul Mediateka**

Modul Mediateka je nadgradnja klasične fototeke in je osnova za digitalno knjižnico v ustanovi, saj omogoča dokumentiranje in hrambo različnih vrst multimedijskih datotek (fotografije, zvočne in video datoteke).

Tako kot ostali moduli omogoča samostojno uporabo, lahko pa se enostavno poveže z drugimi deli programa.

4. Modul Dokumentacija

Modul Dokumentacija predstavlja celovit sistem za obdelavo podatkov in objavo izbranih podatkov na spletu in v mobilnih aplikacijah.

Omogoča:

- vnos podatkov o razstavah in otvoritvah ter dokumentacije o razstavah;
- vodenje kataloške dejavnosti (publikacije, katalogi, letaki);
- vnos podatkov o člankih;
- objavo razstav in člankov v spletnem sistemu Museums.

5. Pedagoški modul

Pedagoški modul omogoča natančen popis pedagoških aktivnosti, ki jih ponuja muzej ali galerija (delavnice, vodeni ogledi, obiski, …), skupaj z rezervacijami oz. s popisom vseh izvedb (rezervacije terminov, …). Na voljo je tudi grafični koledar, ki lajša načrtovanje izvedb aktivnosti.

Pedagoške aktivnosti je mogoče objaviti tudi v sistem Museums.

6. Modul Knjižnica

Modul Knjižnica je namenjen popisu knjižnega gradiva, ki ga hrani ustanova.

Poleg vnosa gradiva in iskanja po vnesenih zapisih omogoča tudi tiskanje izpisov in beleženje izposojenega gradiva.

7. Modul Arhiv

Modul Arhiv je namenjen popisovanju arhivskega gradiva. Modul je skladen z uveljavljenim Splošnim mednarodnim standardom za arhivsko popisovanje ISAD(G) [12], z drugo izdajo iz leta 2000.

Modul omogoča popisovanje arhivskega gradiva skladno z omenjenim standardom, dodajanje fotografij popisanega gradiva, iskanje po vseh poljih shranjenih zapisov in tiskanje zapisov.

8. Museums

Z izdelavo spletnega portala Spletna galerija (www.spletna-galerija.net) v letu 2007 so kulturne ustanove dobile možnost predstavitve svojega dela – v prvi vrsti zbirk in predmetov

- širši javnosti na spletu. Enostavna spletna stran je omogočala objavo osnovnega nabora informacij o muzealijah iz programa Galis.

Spletna galerija se je v letu 2010 razširila in preoblikovala v Museums.si (www.museums.si), ki predstavlja slovenske muzeje in galerije. Podatke je mogoče na spletu objaviti iz programa Galis ali preko spletnega urejevalnika podatkov, ki je dostopen v spletnem brskalniku z uporabniškim imenom in geslom.

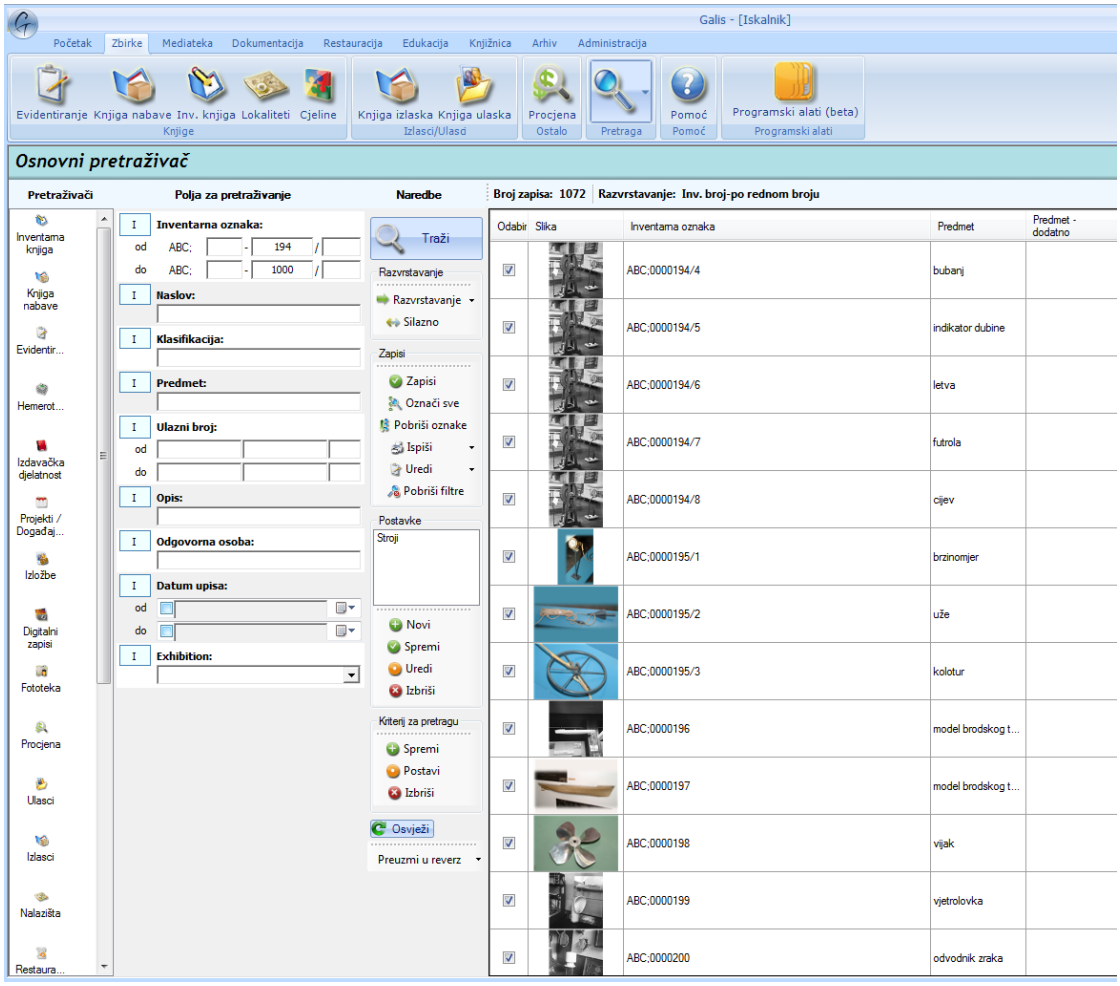
Platforma se je s portaloma Museums of the World (www.museu.ms) in Museums.eu (www.museums.eu) razširila v globalno muzejsko spletno platformo, ki jo za svojo predstavitev uporabljajo kulturne ustanove po vsem svetu. Vključuje tudi predstavitev v povezanih mobilnih aplikacijah za vse najbolj razširjene mobilne platforme (Android in iOS).

Skupne komponente programa Galis

Program Galis vsebuje tudi več skupnih komponent, ki jih uporabljajo vsi moduli programa, za to diplomsko delo sta bistveni predvsem naslednji komponenti:

1. Iskalnik

Iskalnik je z vidika uporabnikov programa Galis ena izmed najpomembnejših funkcionalnosti. Omogoča iskanje po celotni podatkovni bazi in pripravo izpisov, ki jih je mogoče prilagoditi zahtevam uporabnika.

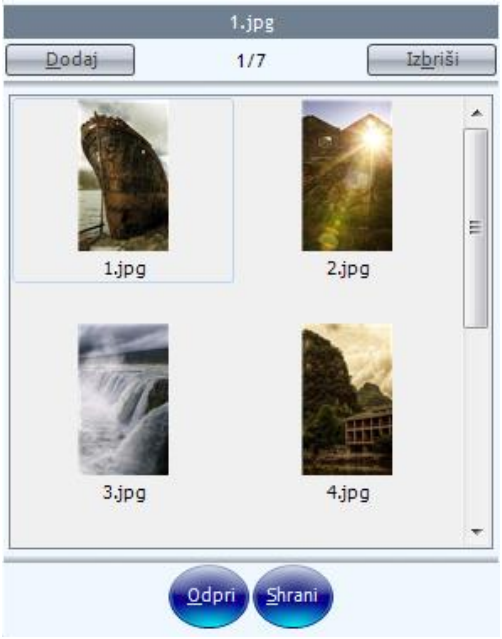


Slika 2.2: Iskalnik v programu Galis

2. Dodajanje/urejanje fotografij

Večina zapisov v podatkovni bazi programa Galis je opremljenih z eno ali več fotografijami, zato je za uporabnike pomembno, da je delo s fotografijami – njihovo dodajanje, urejanje, pregled in iskanje – čim bolj pregledno in enostavno.

Galis omogoča dodajanje fotografij k vsakemu zapisu, pa tudi njihovo urejanje.



Slika 2.3: Dodajanje in pregled fotografij v programu Galis

Pomemben del dodajanja fotografij je možnost dodajanja oznak/značk (*ang. tags*), kjer lahko uporabniki dele fotografij označijo s posebnimi značkami.

Slika 2.4: Označevanje značk na fotografijah

2.2.2 Pregled izvajalnega okolja

Ker diplomsko delo razširja programski paket Galis na kratko povzemamo tudi opis tehnologij, na katerih je Galis zgrajen.

Programski paket je zgrajen s pomočjo Microsoft .NET ogrodja, natančneje njegove različice 3.5., uporablja pa Windows Forms (*tudi WinForms*) tehnologijo kot osnovno tehnologijo za uporabniški vmesnik. Podatkovna baza, ki jo Galis uporablja je Microsoft SQL 2005 ali novejša.

Prvo različico ogrodja (1.0) je Microsoft izdal v začetku leta 2002, trenutno je na razpolago različica 4.6.2, ki je izšla v začetku avgusta 2016. [13] Microsoftovo razvojno okolje (*ang. Integrated Development Environment – IDE*) je Microsoft Visual Studio, trenutna različica je Visual Studio 2015.

.NET Framework 4.6 podpira različne jezike, med najbolj popularne sodita Visual Basic in C#, ki v osnovi temelji na sintaksi jezika C++ [13] in je tudi programski jezik, ki ga uporablja Semantika pri razvoju programskega paketa Galis. Aplikacije, spisane v .NET se prevedejo v vmesni jezik, ki se izvaja znotraj .NET izvajalnega okolja imenovanega CLR (Common Language Runtime). [14]

Za namene te diplomske naloge je pomembno predvsem dejstvo, da gre za t.i. nadzorovan (ang. *managed)* programski jezik, kar pomeni, da izvajalno okolje samo skrbi za sproščanje pomnilnika objektov, ki niso več potrebni s pomočjo smetarja (ang. *garbage collectorja)*. Pomembno je tudi, da jezik omogoča integracijo klasičnih Windows dinamičnih knjižnic (DLL – Dynamic Link Library), spisanih v programskem jeziku C++ z uporabo t.i. *Interop* nivoja. [15]

Takšne (*unmanaged*) knjižnice lahko v aplikacije integriramo na tri načine [15]:

- P/Invoke: mehanizem, ki omogoča, da v nadzorovanem jeziku kot je C# definiramo signaturo metode v nenadzorovanem DLL in jo potem iz nadzorovane kode kličemo enako, kot bi klicali nenadzorovano metodo;
- COM (Component Object Model): tehnologija podjetja Microsoft, ki se opušča in za to diplomsko delo ni zanimiva;
- IJW (It Just Works): dovoljuje uporabo C++ metod neposredno iz nadzorovanega jezika, dokler so izpolnjeni nekateri predpogoji glede tipov vhodov in izhodov in načina gradnje DLL datoteke.

Možnost povezovanja nadzorovane in nenadzorovane kode je za nas pomembna predvsem zato, ker je obdelava slik procesorsko zelo zahtevna. Zato je potrebno določene dele kode spisati v nenadzorovanih jezikih. Določene knjižnice, ki jih opisujemo v poglavju 2.2.4, so prav tako na razpolago zgolj v C++ obliki, zato je enostaven način za povezavo z njimi dobrodošel.

2.2.3 Pregled problemov in rešitev za iskanje in indeksacijo besedil

Preden se lotimo opisa platform, ki omogočajo indeksacijo (in priklic/iskanje) besedil, moramo najprej na kratko definirati tipične probleme, ki jih skušamo rešiti in motivacijo za reševanje le-teh.

Omenili smo že, da je za uporabnike iskanje zelo pomembna funkcionalnost aplikacije, sinonim za iskanje pa je v zadnjih 15 letih postal Google in njegov način iskanja spletnih dokumentov. Eden izmed ciljev je bil torej, da uporabnikom omogočimo iskalno izkušnjo podobno Googlovi, kjer z vnosom ključnih besed dobijo rezultate, ki tem ključnim besedam ustrezajo ne glede na to, kje se te nahajajo.

To je samo del razloga, saj smo želeli zgraditi učinkoviti indeks za vsa polja podatkovne baze z namenom podpore še ene funkcionalnosti. V okviru te diplomske naloge smo namreč razvili tudi funkcionalnost, ki uporabnikom predlaga vrednosti za polja na novih vnosih. S tem lahko namreč precej pohitrimo vnos za uporabnike, pravilno strukturirano kazalo pa je lahko dobra osnova za izgradnjo te funkcionalnosti.

S pojmom indeksacije besedil znotraj tega diplomskega dela mislimo pravzaprav na kombinacijo dveh pristopov k iskanju:

- iskanje po metapodatkih (*ang. metadata based search*);
- iskanje po polnem besedilu (*ang. full text search*).

Spomnimo, da programski paket Galis podatke hrani v podatkovni bazi Microsoft SQL, kar pomeni, da kot podatkovni vir uporablja relacijsko podatkovno bazo. To pomeni, da je iskanje po metapodatkih že mogoče z uporabo SQL (Structured Query Language) poizvedb, natančneje z Microsoft implementacijo SQL imenovano Transact-SQL [16], kar je tudi način, ki ga že podpira iskalnik v programskem paketu Galis, opisan v poglavju 2.2.1. Kadar želimo iskati po numeričnih, datumskih ali drugih preprostih poljih, lahko dosežemo zadovoljive hitrosti iskanja že z uporabo vgrajenih načinov indeksacije v strežnik SQL, ki so v različici MS SQL 2008 R2 naslednje [17] [18]:

- Kazalo z gručami (*ang. clustered index*): interno implementiran kot B-drevo združuje podatke v grozde na podlagi ključa;
- Kazalo brez gruč (*ang nonclustered index*): omogoča, da nad tabelo, ki že ima kazalo z gručami ali kopico, dodamo dodatno kazalo, ki hrani vrednost izbranega polja in kazalec na gručo v kazalu z gručami;

- unikatno kazalo (*ang. unique*): omogoča, da definiramo, da ključ v kazalu ne more vsebovati podvojenih vrednosti in ga je mogoče dodati obema kazaloma opisanima v točkah 1 in 2;
- razširjeno kazalo (*ang. index with included columns*): omogoča, da v kazalo dodamo dodatna polja, ki niso vsebovana v ključu;
- polno besedilno kazalo (*ang. full text index*): omogoča iskanje po polnem besedilu znotraj strežnika SQL – *to nekoliko podrobneje opisujemo v nadaljevanju tega poglavja;*
- prostorsko kazalo (*ang. spatial index*): omogoča iskanje po prostorskih podatkih, npr. po geometrijskih koordinatah;
- filtrirano kazalo (*ang. filtered*): omogoča optimizacijo kazala za vnaprej znano podmnožico podatkov iz tabele;
- specializirano XML kazalo (*ang. XML index*): omogoča iskanje po podatkih shranjenih v polju tipa Xml, ki ga MS SQL neposredno podpira.

Kadar želimo pospešiti iskanje po poljih znotraj baze moramo torej vnaprej pripraviti kazala, ki bodo pohitrila pogoste ali dlje trajajoče poizvedbe. [19]

Nekoliko bolj zapleten problem pa je zagotavljanje hitrega iskanja po besedilnih poljih, saj se tukaj srečamo še z nekaj dodatnimi težavami:

- ista beseda se lahko pojavlja v več oblikah, navadno želimo iskati po vseh (iskalna poizvedba »slika« naj najde tudi zadetke, ki vsebujejo: »slike«, »sliko«, »sliki«, …);
- iskanje delov besed je časovno lahko zelo potratno, saj pomeni, da moramo preiskati vse kombinacije, da vidimo, če katera ustreza.

Hitro vidimo, da je potrebno v primeru besedilnega iskanja nekoliko prilagoditi pristop, da bomo dosegli zadovoljive rezultate. Pogosto uporabljen pristop je, da problem razdelimo v dva dela [20] [21]:

- indeksiranje dokumentov z normalizacijo in razbijanjem na žetone,
- iskanje z normalizacijo in razbijanjem poizvedbe.

Osnovne funkcionalnosti polnega besedilnega iskanje podpira že MS SQL strežnik [18], vendar pa je nastavljivost vgrajenega načina relativno slaba in neprilagodljiva, prav tako je zelo omejena podpora za slovenski jezik. Iz tega razloga smo preverili, katere rešitve za polno besedilno iskanje na trgu že obstajajo, našli smo naslednje večkrat izpostavljene rešitve [20] [22]:

- Lucene [23],

- Solr [24],
- Elasticsearch [25],
- ClusterPoint Server [26],
- Azure Search [27].

Podrobnejša analiza naštetih rešitev je pokazala, da jih veliko temelji na Apache Lucene, zato smo se odločili, da podrobneje preučimo primernost le-te za uporabo v našem primeru. [21] [28] [24] [25] [26] [27]

Apache Lucene je zgrajen v javanski tehnologiji, trenutna različica je 6.1.0 [28].Glavne značilnosti strežnika glede na spletno stran pa so:

- prepustnost gradnje kazala pribl 150GB/uro na moderni strojni opremi;
- velikost kazala med 20% in 30% velikosti izvornih podatkov;
- možnost lastnih rangirnih funkci;j
- možnost sortiranja po vseh poljih;
- možnost hkratne gradnje kazala in iskanja;
- iskanje po posameznih poljih;
- dinamično generiranje filtrov (*ang. facets*).

Ker je Lucene licenciran z Apache 2.0. licenco, je združljiv z uporabo v komercialni programski opremi brez posebnih omejitev in se je torej izkazal kot primerna izbira za gradnjo kazala in za izvajanje iskanja ter primerna osnova funkcionalnosti predlogov vnosov za polja. To funkcionalnost podrobneje opisujemo v poglavju 3.2.

Kljub temu da se je Lucene izkazal kot primerna osnovna platforma, je težava v tem, da je razvit v programskem jeziku Java, medtem ko naša rešitev temelji na Microsoft .NET platformi. Rešitev te težave smo našli v paketu FlexLucene [29], ki je različica Lucene, zgrajen je s pomočjo platforme IKVM.NET [30].

IKVM.NET je popolna implementacija javanskega virtualnega stroja (*ang. Java Virtual Machine - JVM*) in omogoča izvajanje javanske kode neposredno v Microsoft .NET CLR okolju.

2.2.4 Pregled problemov in rešitev za računalniški vid

Cilj diplomske naloge je bil razviti tudi modul, ki bo omogočal samodejno segmentacijo slik in razpoznavanje zanimivih objektov na slikah ter kasnejše iskanje slik na podlagi na njih zaznanih zanimivih objektov.

V idealnem primeru smo želeli funkcionalnost dodajanja slik v programskem paketu Galis (opisano v poglavju 2.2.1) dopolniti tako, da bi proces dodajanja slike vključeval:

- nalaganje slike v Galis,
- samodejno segmentacijo slike in razpoznavo objektov,
- dodajanje slike v iskalno kazalo, opisano v poglavjih 2.2.3 in 3.2.

S tehničnega vidika moramo rešiti s področja računalniškega vida torej dva ločena problema:

- segmentacijo slike,
- razpoznavo objektov na sliki

Segmentacija slik

V okviru diplomske naloge smo razvijali splošno-namensko aplikacijo za segmentacijo, ki na neznanih slikah poišče vnaprej definirane razrede znanih objektov.

Čeprav je deljenje slik (ali drugače: ekstrakcija zanimivih območji) za ljudi relativno enostaven proces, sodi med težavnejše procese za računalniško obdelavo. Ljudje namreč pri razpoznavanju slik uporabljamo veliko količino predhodnega poznavanja oblik predmetov, ki je vsaj v tem trenutku še ni mogoče učinkovito digitalizirati [31], hkrati si ljudje pri razpoznavanju objektov velikokrat pomagamo tudi z zaznanim kontekstom objektov in drugimi informacijami, ki jih imamo o videni sliki. [32]

Dodatna težava pri segmentaciji slik je relativna časovna zahtevnost algoritmov za segmentacijo, ki je odvisna od velikosti vhodne slike in njene barvne globine, kar je lahko težava predvsem pri obdelavi večjih količin slik naenkrat.

Osnovni cilj vseh metod segmentacije slike je, da na sliki najdejo množico vsebinsko povezanih slikovnih točk (pikslov).Segmente lahko kasneje nadalje obdelamo glede na naše potrebe. Ker je področje segmentacije slik hitro se razvijajoče področje, na katerem ne obstaja enotna teorija, obstajajo zelo različne klasifikacije algoritmov za segmentacijo [33]. Ena izmed delitev, ki se pojavlja v strokovni literaturi [33] pozna naslednje osnovne vrste algoritmov:

- pragovne algoritme,
- algoritme na podlagi rasterizacije (*ang pixel based*),
- segmentacija na podlagi prostorskih območji,
- segmentacija na podlagi barvnih območji / barvne podobnosti,

- segmentacija na podlagi teorije *fuzzy* množic.

Pri izbiri algoritma za segmentacijo navadno upoštevamo več faktorjev, čeprav lahko ponovno ugotovimo, da ne obstaja splošno priznana objektivna metrika za izbiro primernega delitvenega algoritma [34].

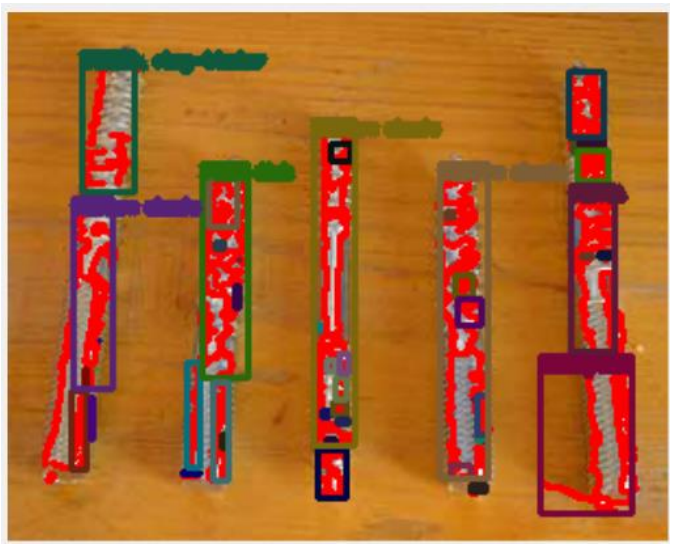
V okviru diplomske naloge smo tako sami definirali faktorje, ki so za nas pomembni pri izbiri algoritma:

- zahtevnost implementacije,
- časovna zahtevnost algoritma,
- pravilnost razpoznanih objektov na omejenem naboru primerov.

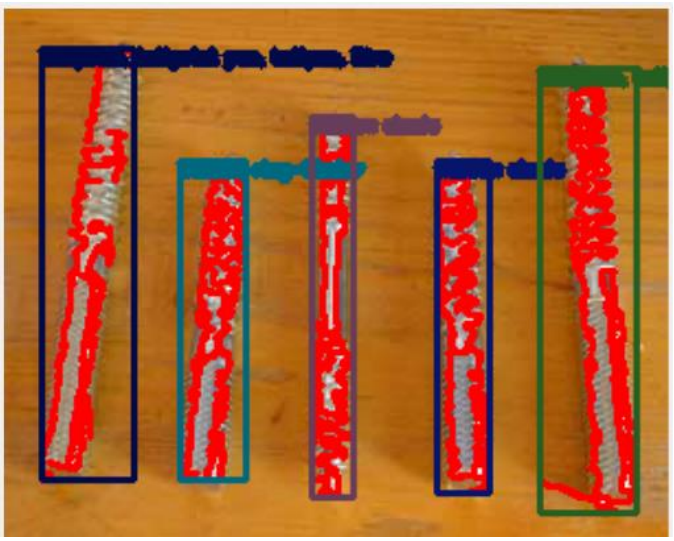
Zahtevnost implementacije je za nas pomembna predvsem zaradi težavnosti vzdrževanja algoritma in posledično dolgoročnih stroškov vzdrževanja razvite rešitve.

Časovna zahtevnost izvajanja algoritma je v našem primeru precej pomembna, saj želimo slike obdelovati »v realnem času«, to pomeni, da želimo uporabniku predlagati zanimive objekte takoj zatem, ko sliko doda v aplikacijo. Želimo si torej, da je celotno razpoznavanje območji izvedeno v največ nekaj sekundah.

Pomembna metrika je seveda tudi pravilnost zaznanih območij, kar pomeni, da naš algoritem zazna vse objekte, ki nas zanimajo in da vsakega izmed njih razpozna kot enotni objekt. Primer slabe in dobre segmentacije prikazujemo na sliki 2.5 in 2.6; na obeh slikah je uporabljen isti algoritem z različnimi parametri za segmentacijo. Prikaz neobdelane slike je na sliki 2.7.



Slika 2.5: Primer pretirane segmentacije slike



Slika 2.6: Primer dobre segmentacije slike

Slika 2.7: Slika, ki smo jo segmentirali na prejšnjih slikah

Ker predpostavljamo, da bomo našo aplikacijo uporabljali v relativno dobro znanih pogojih (ali drugače: vemo, kakšne kategorije objektov bomo razpoznavali), je te pogoje mogoče preveriti empirično in primerno posodobiti parametre algoritma, če na začetku izbrani parametri ne dajejo primernih rezultatov za izbrani nabor razpoznav (glej tudi poglavje 3.2).

Razpoznavanje zanimivih objektov

Ko smo rešili problem segmentacije slike, imamo sliko razbito na več manjših sličic, vendar pa še zmeraj ne vemo, katera izmed njih vsebuje za nas zanimive objekte. Zato moramo sedaj vsak posamezni objekt še identificirati, kjer gre pravzaprav za še težji problem kot problem segmentacije slik, ki smo ga že opisali v poglavju »Segmentacija slik«.

Večinoma k problemu razpoznavanja slik pristopamo z uporabo različnih metod strojnega učenja, med bolj znanimi metodami razpoznavanja je metoda nevronskih mrež [35] [36]. Pri tem se je v zadnjih petih letih na več področjih računalniškega vida uveljavil pristop uporabe t.i. globokih nevronskih mrež (*ang. DNN -Deep Neural Networks*), npr. GoogleNet mreže [36], ki je vrsta konvolucijske nevronske mreže. [37]

Težava nevronskih mrež, ki dosegajo dobre rezultate, je predvsem v tem, da njihovo učenje traja veliko časa (nekaj tednov na modernih GPU podprtih računalniških grozdih) [38], zato je lahko tako učenje v praksi težko izvedljivo z omejenimi viri. Alternativni pristop je torej, da uporabimo vnaprej naučeno nevronsko mrežo, ki je že naučena za razpoznavanje širokega nabora objektov in jo potem do-naučimo (*ang fine-tuning*) na našem vzorcu. [39] [40] Primer razpoznave slike z nevronsko mrežo prikazujemo na sliki 2.8.

Slika 2.8: Primer razpoznave slike pred optimizacijo parametrov

Težava globokih nevronskih mrež v našem kontekstu je tudi relativno dolg čas razpoznave objektov (t.i. forward pass), ki pa ga lahko s primernimi metodami strojne optimizacije precej skrajšamo. Tako lahko na primer dosežemo še sprejemljive čase z uporabo grafične procesne enote (GPU) pri procesiranju podatkov. Pri tem sta uporabni predvsem dve

ogrodji, ki omogočata izkoriščanje grafične kartice za uporabo pri treniranju mrež: Cuda, ki je tehnologija podjetja NVidia, in OpenCL, ki je odprt standard, ki ga podpirata npr. Intel in AMD.

Optimizacija parametrov algoritma

Kot smo omenili v poglavju »Razpoznavanje zanimivih objektov« je količino zaznanih objektov in združevanja mogoče upravljati, vendar pa je primerno optimizirane parametre na roko praktično nemogoče določiti, saj imamo na razpolago več parametrov, kjer lahko vsak zavzame širok nabor vrednosti. Eden izmed pristopov za avtomatsko optimizacijo takih parametrov je pristop z uporabo genetskih algoritmov, kjer na primer [41] [42]:

- naključno generiramo množico kombinacij parametrov;
- na učni množici preizkusimo, kakšne rezultate dobimo;
- izberemo določen odstotek najboljših iz množice generiranih parametrov;
- z izbrano funkcijo križamo izbrane, da dobimo novo generacijo;
- določen odstotek naključno mutiramo, da se izognemo lokalnim optimumom;
- postopek ponavljamo.

Na ta način po dovolj ponavljanjih dobimo parametre, ki minimizirajo napako (fitnes funkcijo) in se čim bolj približajo želenim vzorcem razpoznavanja.

Izbira knjižnice za računalniški vid

Ko smo na kratko preverili za nas relevantne osnovne probleme računalniškega vida in pogledali nekaj pristopov k reševanju, smo pregledali še knjižnice, ki omogočajo enostavnejšo implementacijo rešitev s področja računalniškega vida in so kompatibilne z našo izbrano platformo (.NET, C#). V poglavju 2.2.2 smo omenili, da lahko koda C# preko različnih mehanizmov kliče »klasično« C++ kodo in omenili, da je zaradi hitrosti izvajanja zaželeno, da so vsaj kritični deli kode napisani v tem programskem jeziku. V poglavju 2.2.4, podpoglavju »Razpoznavanje zanimivih objektov« smo nato poudarili, da lahko pri določenih operacijah dosežemo precejšnje pohitritve (tudi 10x - 20x), če uporabimo grafične procesorje za izvajanje določenih matematičnih operacij (npr. matrično množenje). [5] [6]

Glede na zahteve segmentacije smo si pred izbiro postavili naslednje kriterije:

- knjižnica je implementirana v C++ ali C#, podpora za CUDA in OpenCL je prednost;
- knjižnica ima vgrajeno podporo za branje pogostih slikovnih formatov;
- knjižnica ima podporo za osnovne filtrirne operacije nad slikami;

- knjižnica ima podporo za izvajanje (*t.j. forward-pass*) naučenih nevronskih mrež
    - podporo za DNN,
    - podporo za ConvNet mreže.

Knjižnica OpenCV je izbranim pogojem najbolj zadoščala. Sredi leta 2000 jo je predstavil Intel, v letu 2012 pa je razvoj prevzela neprofitna organizacija OpenCV, ki ji je Intel podaril izvorno kodo. [43]

Pred uporabo smo seveda preverili licenčno skladnost knjižnice s cilji projekta. Ker je knjižnica izdana pod 3-točkovno BSD licenco (*ang. 3-clause BSD license*), je skladna in smo jo lahko izbrali za uporabo v naši rešitvi. [44]

Knjižnica je izvedena v programskem jeziku C++.

Izbira knjižnice za učenje nevronskih mrež

V predhodnih podpoglavjih smo izbrali in utemeljili izbiro globokih nevronskih mrež, kot osnovno tehnologijo za razpoznavanje objektov, ugotovili pa smo tudi, da je učenje take mreže precej zahtevno, če želimo začeti iz »prazne« mreže in praktično nedosegljivo za manjša podjetja, ki nimajo zahtevne računalniške infrastrukture.

Na drugi strani pa se je v zadnjih letih pojavilo kar nekaj ogrodji za globoko učenje, med katerimi je potrebno v okolju C++ ob upoštevanju licenčnih omejitev omeniti vsaj [45]:

- CNTK,
- OpenNN,
- Theano,
- TensorFlow,
- Torch,
- Caffe,

Ker smo se pri knjižnici za izvedbo razpoznavanja vida že odločili za OpenCV (*glej prejšnje poglavje*), je bila najbolj smiselna izbira ogrodje Caffe iz naslednjih razlogov [46]:

- Caffe interno kot osnovo uporablja OpenCV;
- OpenCV ima DNN modul, ki podpira obliko datotek, ki jih pri treningu generira Caffe;
- za Caffe obstajajo vnaprej naučeni modeli za nevronske mreže brez omejitev za komercialno uporabo, ki jih najdemo v »Model Zoo« [47].

Kot osnovo platformo za nevronske mreže smo na koncu torej izbrali Caffe, kot osnovno nevronsko mrežo za dodatno učenje (*fine tuning*) pa vnaprej naučen GoogleNet iz »Model Zoo.« [47]

3 ZASNOVA IN IMPLEMENTACIJA REŠITVE

*»**Clarke's three laws***

- *When a distinguished but elderly scientist states that something is possible, he is almost certainly right. When he states that something is impossible, he is very probably wrong.*
- *The only way of discovering the limits of the possible is to venture a little way past them into the impossible.*
- *Any sufficiently advanced technology is indistinguishable from magic.*[3]*«* [48]

3.1 Arhitektura rešitve

Preden opišemo arhitekturo še enkrat povzemimo tehnološke rešitve, ki smo jih izbrali v poglavju 2:

- Osnovna platforma: Microsoft .NET;
- Knjižnica za polno besedilno iskanje: Lucene;
- Knjižnica za računalniški vid: OpenCV.

Poglejmo sedaj še nekaj implementacijskih podrobnosti, ki jih v okviru poglavja 2 nismo posebej obdelali, so pa pomembne za izvedbo delujočega produkta, ki ga lahko integriramo v programski paket Galis in so primerne za uporabo v produkciji.

Osnovno platformo smo izbrali, zato smo se morali odločiti še za izbiro natančne tehnologije in načina izvedbe, ki ju želimo uporabiti za implementacijo projekta diplomskega dela.

---

[3]*Prevod: Trije zakoni Clarka*

Če starejši uveljavljen znanstvenik reče, da je nekaj mogoče, ima skoraj zagotovo prav. Če reče, da je nekaj nemogoče, se verjetno moti.

Edini način, da spoznamo meje mogočega je, da gremo malo čez v domeno nemogočega.

Dovolj napredne tehnologije ne moremo ločiti od magije.

Spomnimo se, da je Galis zgrajen na Windows Forms tehnologiji, zato bi najprej pomislili na neposredno integracijo rešitve v aplikacijo.

Vendar pa lahko znotraj iste organizacije več uporabnikov na različnih računalnikih hkrati uporablja Galis, zato bi v primeru takega načina integracije morali zagotoviti nek mehanizem, preko katerega bi posodabljali zgrajeno kazalo za iskanje po polnem besedilu – zato moramo predlagati rešitev, ki bo omogočala uporabo več računalnikom hkrati. Ker je indeks lahko tudi relativno velik, želimo tudi, da ga hranimo samo na eni lokaciji.

Dobra izbira glede na podane pogoje, je torej izbira spletne rešitve, ki jo znotraj organizacije namestimo enkrat, do nje pa dostopajo vsi uporabniki Galisa znotraj organizacije. Ker smo izbrali platformo .NET, se moramo odločiti za eno izmed podprtih tehnologij za spletne storitve [49]:

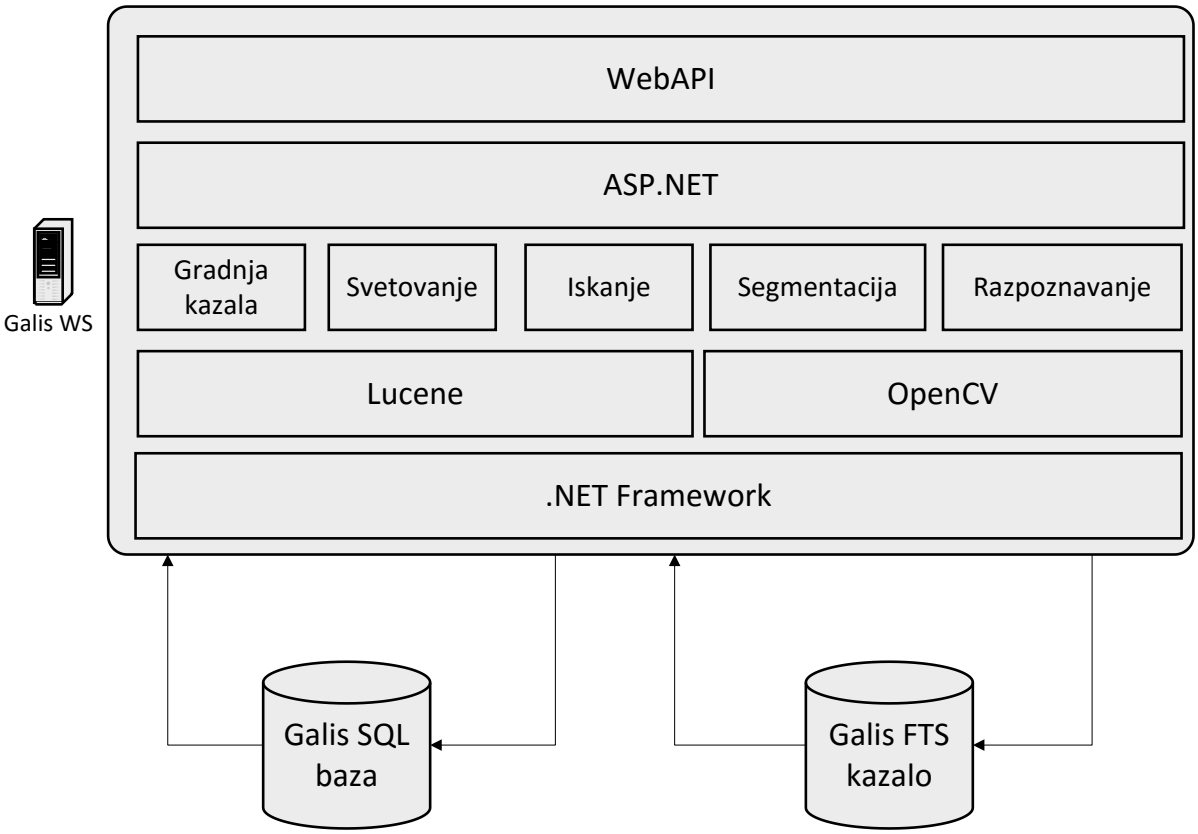
- ASP.NET + Windows Communication Foundation,
- ASP.NET + WebAPI.

Glede na moderne smernice razvoja spletnih storitev smo se odločili za razvoj REST (REpresentational State Transfer) spletnih storitev, glede na trende v razvoj Microsoft .NET platforme pa smo izbrali implementacijo s pomočjo WebAPI, različice 2. [50]

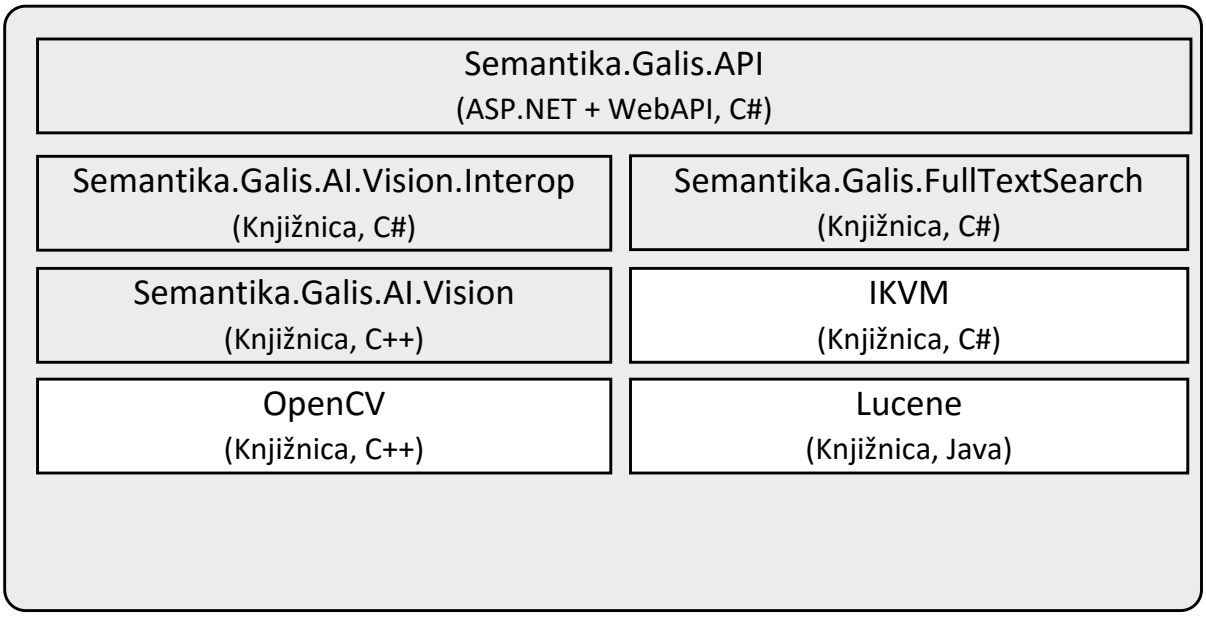
Sedaj moramo v rešitev umestiti obe izbrani platformi: OpenCV in Lucene. V poglavju 2.2 smo navedli, da je OpenCV spisan v programskem jeziku C++, Lucene pa v programskem jeziku Java. Omenili smo tudi, da za Lucene že obstaja vnaprej pripravljen paket FlexLucene, ki ga lahko neposredno vključimo v projekt, ki se izvaja znotraj CLR, [29] zato moramo pogledati samo še, kako smo integrirali knjižnico OpenCV.

V poglavju 2.2.2 smo omenili, da lahko v izvajalnem okolju CLR izvajamo tudi nenadzorovano programsko kodo z uporabo enega izmed treh mehanizmov. V našem primeru smo se odločili, da bomo izdelali vmesni nivo z uporabo P/Invoke metode integracije in da bomo algoritme za segmentacijo in razpoznavanje izvedli s pomočjo C++ kode, ki jo bomo samo poklicali iz skupne rešitve.

Arhitektura tako zasnovane rešitve je predstavljena na sliki 3.1, dejanske komponente pa na sliki 3.2.



Slika 3.1: Arhitektura zasnovane rešitve



Slika 3.2: Pogled arhitekture po komponentah. Z belo so označene zunanje knjižnice, s sivo komponente razvite v okviru diplomskega dela

3.2 Zasnova ključnih algoritmov

Ko smo enkrat zasnovali osnovno arhitekturo rešitve in izbrali platforme za izvedbo ključnih funkcionalnosti, je bilo potrebno razviti delujočo rešitev, ki jo je mogoče integrirati v programski paket Galis.

Ker bi opis razvoja podporne infrastrukture in izvedbe spletnih storitev presegel obseg te diplomske naloge, se v nadaljevanju osredotočamo predvsem na ključne algoritme, ki rešujejo že opisane probleme in so v času razvoja predstavljali večje inženirske izzive. V nadaljevanju zato opisujemo princip delovanja naslednjih rešitev znotraj razvitega produkta:

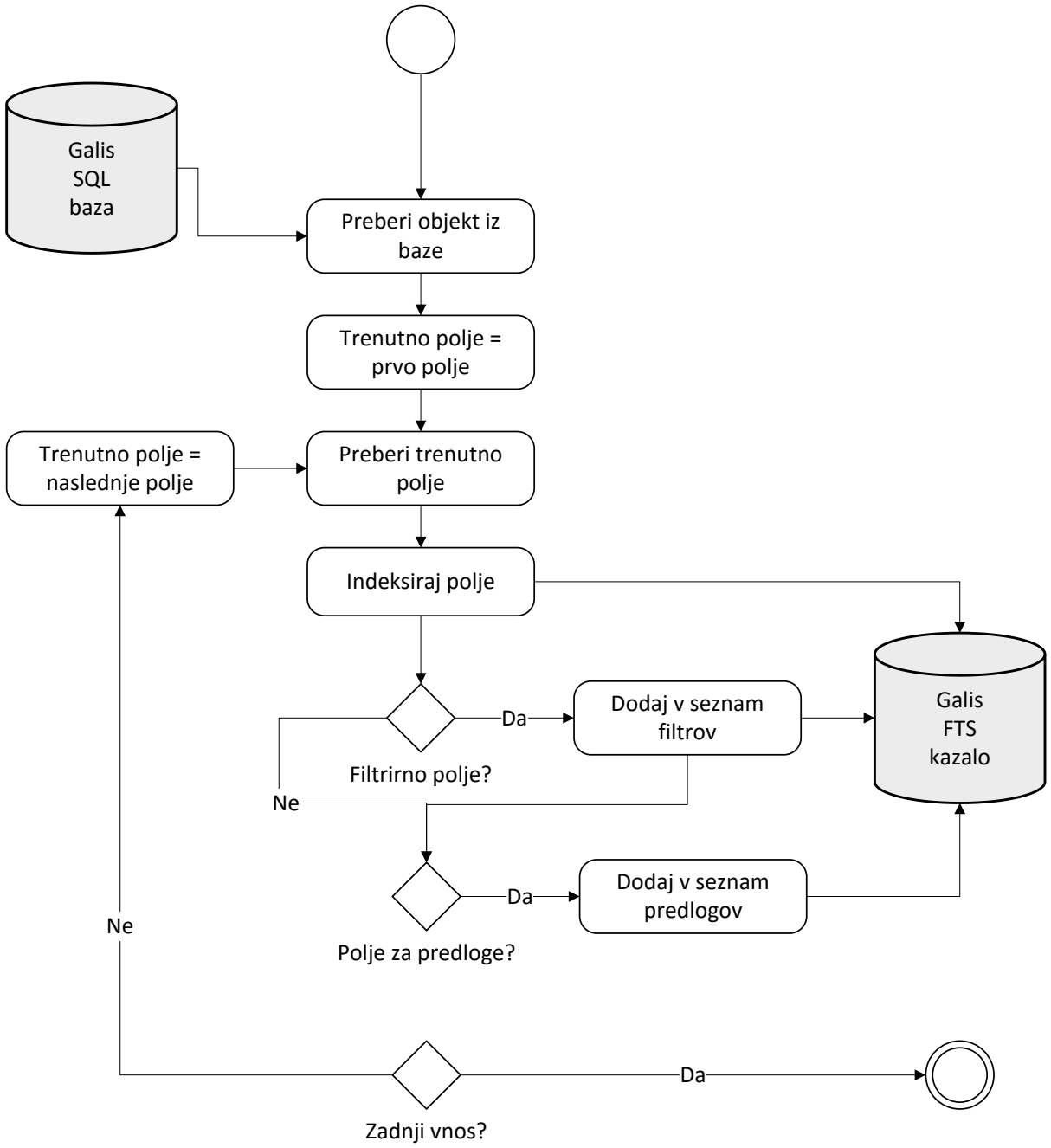
- generiranje kazala, ki ga uporabljamo za polno besedilno iskanje in pripravo predlogov;
- izvajanje poizvedb;
- priprava predlogov za vrednosti polj;
- izvedba segmentacije slik z razpoznavanjem zanimivih predmetov.

3.2.1 Generiranje kazala za iskanje po polnem besedilu

Za iskanje po polnem besedilu smo izbrali knjižnico Lucene, ki smo jo na kratko opisali že v poglavju 2.2.3. Poglejmo sedaj natančneje funkcionalnosti knjižnice, pomembne za generiranje kazala [23]:

- možnost indeksiranja polj numeričnih in tekstovnih oblik;
- možnost indeksiranja več vrednosti za eno polje;
- implementacija podatkovnih struktur za kasnejše hitro iskanje za večdimenzionalna numerična polja (KD-drevesa);
- analizator besedila, ki zna:
    - besedilo razbiti na besede (*ang. tokenization*),
    - koreniti besede;
- gradnje stranskega kazala (*ang. sidecar index*) za filtriranje po poljih.

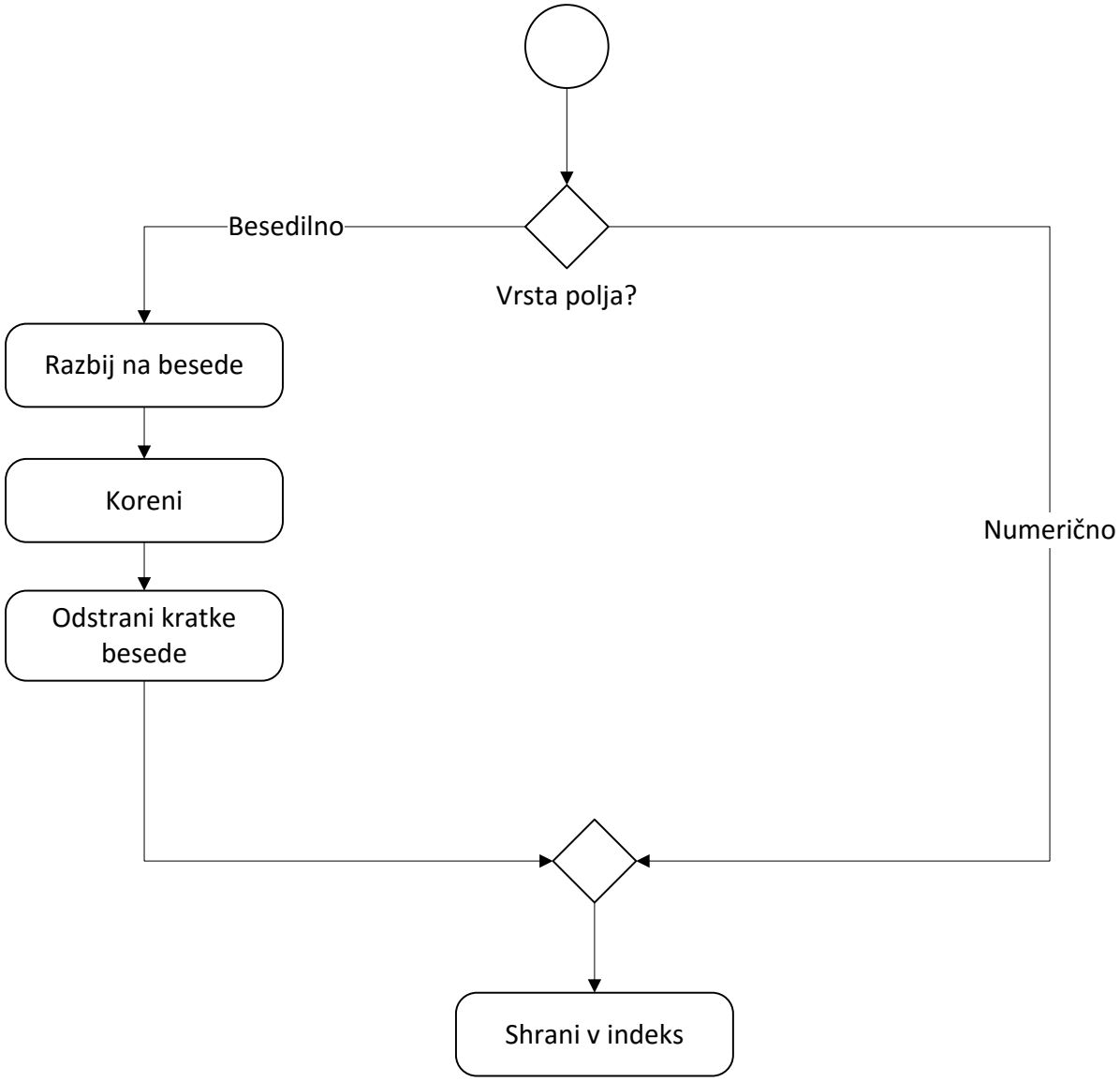
Glede na navedeno smo zasnovali proces gradnje kazala prikazan na sliki 3.3.



Slika 3.3: Proces gradnje kazala za en dokument iz baze

Proces gradnje poteka tako, da preberemo vse dokumente iz baze drugega za drugim in nad njimi izvajamo indeksiranje. Le-to poteka tako, da najprej preberemo prvo polje, ga dodamo v indeks in preverimo, ali ga moramo dodati tudi med filtrirna polja (*facets*) in med polja za generiranje predlogov vrednosti pri vnosu. Ta postopek ponovimo za vsa polja na dokumentu, nato pa se prestavimo na naslednji dokument.

Znotraj samega postopka indeksacije uporabljamo Lucenove možnosti za gradnjo indeksa: to pomeni, da tekstovna polja razbijemo na posamezne besede, ki jih korenimo, nato pa shranimo v index. Če naletimo na numerična polja (kamor sodijo tudi datumska polja), postopamo podobno, le da jih ne analiziramo s tekstovnimi analizatorji. [51] Zelo poenostavljeno shemo indeksiranja posameznega polja prikazuje slika 3.4, natančno je proces opisan v [52]



Slika 3.4: Poenostavljen proces indeksiranja enega polja

3.2.2 Iskanje

Poglejmo še funkcionalnosti knjižnice Lucene, pomembne za izvajanje iskanja [23]:

- izračun podobnosti med dokumenti;
- izračun ujemanja dokumenta z iskalnimi pojmi z uporabo TF-IDF (*ang. za Term Frequency – Inverse Document Frequency*) in izpeljank [53];
- možnost iskanja po vrednosti polj/metapodatkov;
- možnost iskanja v območjih za numerična polja;
- pridobivanje dodatnih filtrov skupaj s številom dokumentov, ki filtru ustrezajo (*facets*).

To pomeni, da lahko na podlagi zgrajenega indeksa s pomočjo knjižnice Lucene iskanje izvedemo relativno preprosto in nazaj dobimo tako iskalne rezultate, ki ustrezajo posredovani poizvedbi, kot tudi seznam filtrov, s katerimi lahko rezultate dodatno filtriramo, skupaj s številom zadetkov, ki jih bomo v primeru uporabe filtra dobili.

Lucene podobnost z dokumenti računa na podlagi TF-IDF metodologije [54], pri čemer najprej upošteva klasično Boolovo filtriranje. Poenostavljeno to pomeni, da iskalnik najprej izloči vse dokumente, ki ne ustrezajo iskalnim pojmom, nato pa za njih izračuna, koliko ustrezajo podani iskalni poizvedbi, da lahko dokumente rangira. TF-IDF za izračun uporablja dve vrednosti [53] [54]:

- pogostost pojava iskalnega izraza v dokumentu (*ang. term frequency*), ki pove, kolikokrat se določen pojem pojavi v posameznem dokumentu;
- obratno pogostost pojava v dokumentih (*ang. inverse document frequency*), ki pove, kako pogosto se določen pojem pojavlja v vseh dokumentih.

TF-IDF deluje torej tako, da pojavnost izraza v dokumentu uteži s pojavnostjo izraza v vseh dokumentih (ali bolj točno: obratno vrednostjo pojavnosti izraza v vseh dokumentih). Na ta način dobijo besede, ki so uporabljene v poizvedbi in so v dokumentih zelo pogoste manjšo težo kot tiste, ki so v dokumentih redkejše in zato bolje določajo posamezni dokument.

3.2.3 Predlogi za vrednosti polj

V poglavjih 3.2.1 in 3.2.2 smo opisali osnovni pristop h gradnji iskalnega kazala in iskanju po kazalu. Ta koncept lahko sedaj nekoliko razširimo in s tem že dobimo enostavno rešitev za enega izmed ciljev te diplomske naloge: pripravo predlogov vrednosti za nove vnose, na podlagi ene vnesene vrednosti v vnosno masko.

Če ponovimo konkretni cilj: uporabnik v vmesniku izbere, da dodaja nov predmet v zbirko predmetov z imenom »Fotografska zbirka Janeza Novaka«. Želimo ugotoviti, kaj so verjetni

vnosi za nekatere druga polja, npr. avtorja. Na podlagi obstoječih vnosov v podatkovni bazi lahko ugotovimo, da imajo vsi vnosi v tej zbirki med avtorji navedenega avtorja »Janez Novak«. Tako lahko z visoko stopnjo verjetnosti trdimo, da bo tudi nov zapis v zbirki imel ta vnos.

Praktično lahko to sedaj implementiramo tako, da:

1. poiščemo vse vnose v kazalu, ki ustrezajo podanemu iskalnemu polju;
2. pridobimo dodatna filtrirna polja iz indeksa skupaj s številom zadetkov znotraj zbirke, ki ustrezajo dodatnemu filtru;
3. izračunamo razmerje med iskalnimi zadetki, ki ustrezajo in vsemi zadetki znotraj zbirke;
4. vrnemo samo tiste filtre, ki ustrezajo minimalnemu pragu, recimo pojavnosti v vsaj 50% dokumentih glede na iskalni pojem.

3.2.4 Segmentacija slik

Kot smo opisali v poglavju 2.2.4 smo morali najprej izbrati algoritem za segmentacijo slike, kjer smo pri izbiri algoritma upoštevali predvsem že omenjene kriterije: pričakovano časovno zahtevnost algoritma in zahtevnost implementacije. Pri tem smo bili pozorni, da algoritem čim bolj parametriziramo in na ta način zagotovimo zadostno nastavljivost, da bomo lahko dosegli pričakovan tretji parameter algoritma, t.j. pravilnost segmentacije objektov.

Da bi dosegli čim boljšo razpoznavanje objektov, smo se odločili za zaznavanje objektov s pomočjo metode zaznavanja robov. V grobem metoda zaznavanja opravi naslednje korake:

- normalizacija (pred-obdelava) slike,
- zaznavanje robov,
- odpravljanje anomalij,
- iskanje objektov in njihovih omejujočih pravokotnikov,
- združevanje in izbris nezanimivih regij.

V poglavju 2.2.4 smo že pogledali osnovne strategije pred-obdelave in zaznavanja robov, zato se tukaj nekoliko bolj posvetimo izbiri praktičnih algoritmov za posamezne korake. Na podlagi omenjenih glavnih delov algoritma smo zasnovali več korakov, ki sliko obdelajo:

1. **Pretvorba barvne slike v sivinsko sliko** z namenom zmanjšanja števila dimenzij (3D -> 1D);

2. **Izvedba bilateralnega filtra z okolico *x*, ter sigma okolico *y* za barvo in razdaljo;** na ta način ohranimo robove, hkrati pa se znebimo manjšega šuma na sliki, kar je v primeru segmentacije na podlagi robov veliko bolj zaželeno, kot klasično linearno filtriranje [55];

3. **Uporaba median filtra z jedrom/oknom velikosti *x*** (*ker delamo s sivinsko sliko,lahko filter izvajamo samo nad enim kanalom*); na ta način se znebimo večjega šuma na sliki in hkrati nekoliko žrtvujemo jasnost robov;

4. **Izvedemo morfološko *odpiranje* in ga *x*-krat ponovim ponovimo**; na ta način zagotovimo, da iz slike odstranimo manjše objekte/robove, ki jih obravnavamo kot šum;

5. **Izvedemo morfološko *zapiranje* in ga *x*-krat ponovimo;** na ta način zapremo manjše odprtine – združimo regije z »luknjami«;

6. **Izračunamo primerno robno točko (ang. *threshold*) za uporabo v zaznavanju robov;** le-to izračunamo z uporabo metode Otsu, ki deluje dobro na sivinskih slikah, ki imajo relativno različne barve ozadja in objektov [56];

7. **Izvedemo Canny** [57] **zaznavanje robov,** kjer kot visoko vrednost praga uporabimo izračunano vrednost iz točke 5 in kot nizko vrednost, vrednost visoke vrednosti, pomnoženo s faktorjem *x;*

8. **Izvedemo dilacijo;** na ta način zapremo manjše odprtine, ki nastanejo pri zaznavanju robov;

9. **Izvedemo poligonsko aproksimacijo robov po algoritmu Abe in poenostavimo polinome z uporabo algoritma Douglas-Peucker** [58] [59]**;** na ta način dobimo geometrijske objekte, ki jih lahko z enostavnimi algoritmi iskanja presekov združimo, če so dovolj blizu;

10. **Združimo poligone** [60]**, ki so dovolj blizu in izračunamo minimalni pravokotnik, ki vsebuje vsakega izmed poligonov.**

Iz gornjega opisa algoritma je razvidno, da je mogoče algoritem deloma upravljati in s tem določati, kako se bo segmentacija izvedla z naslednjimi parametri:

- okolici *x* in *y* za bilateralni filter iz točke 2;
- velikost okna median filtra *x* iz točke 3;
- število morfoloških odpiranj *x* iz točke 4;
- število morfoloških zapiranj x iz točke 5;
- faktor, s katerim pomnožimo izračun praga za visoko vrednost, da dobimo prag za nizko vrednost iz točke 7;
- faktor prekrivanja dveh pravokotnikov, da ju združimo v točki 10;
- faktor zahtevane barvne podobnosti dveh pravokotnikov, da ju združimo v točki 10;

- najmanjši dovoljeni pravokotnik zaznanega objekta, da ga ne izbrišemo v točki 10;
- največji dovoljeni pravokotnik zaznanega objekta, da ga ne izbrišemo v točki 10.

Kot je razvidno iz slik 2.5 in 2.6 lahko s temi 10 parametri precej dobro upravljamo karakteristike zaznavanja glede na pričakovane vhode.

V poglavju 2.2.4 smo omenili, da smo rešitev zgradili na podlagi knjižnice OpenCV, ki za nas že implementira nekatere algoritme, naštete v točkah 1 – 10:

- pretvorba slike,
- bilateralni, median in Gauss filter,
- osnovne morfološke operacije,
- računanje praga po metodi OTSU,
- sledenje robov Canny,
- algoritem Abe,
- algoritem Douglas-Peucker.

V naši rešitvi smo morali torej pravilno izbrati jedra za izvedbo filtriranj, združiti operacije v pravilno zaporedje in izvesti združevanje in brisanje zaznanih regij.

3.2.5 Razpoznavanje zanimivih objektov

Ko dobimo seznam razpoznanih objektov, moramo vsakega izmed njih še identificirati. V ta namen smo izbrali tehnologijo globoke nevronske mreže (DNN), kjer smo kot osnovo vzeli vnaprej naučeno nevronsko mrežo GoogleNet [61], ki smo jo dodatno naučili v 500 iteracijah z naključnimi slikami iz podatkovnih zbirk podjetja Semantika.

Razpoznavanje posameznega objekta poteka v naslednjih korakih:

- normalizacija velikosti na velikost 224x224,
- sliko po večji stranici zmanjšamo na 224px,
- po manjši stranici jo po potrebi centriramo,
- zapolnimo prazni prostor s črnino (0 na vseh barvnih kanalih),
- centriranje slike (odštevanje srednje vrednosti po vseh kanalih),
- procesiranje slike z nevronsko mrežo (*forward pass*).

3.2.6 Optimizacija parametrov na učni množici

V poglavju 2.2.4 smo omenili, da je izbira pravilnih parametrov eden večjih problemov. Sedaj, ko smo nekoliko natančneje opisali algoritem, je to še veliko bolj očitno, saj imamo 10 parametrov algoritma iz domene realnih števil, torej imamo neskončno možnih kombinacij – ročna optimizacija zato nikakor ni mogoča.

Zato smo za optimizacijo izbrali genetski algoritem, opisan v poglavju 2.2.4, kjer smo izbrali naslednje parametre:

- generacijo velikosti 50 ;
- fitnes funkcija je bila vsota ploščin pravilno razpoznanih segmentov, od katerih smo odšteli ploščine napačno razpoznanih. Fitnes funkcijo smo maksimirali;
- mutirali smo naključno med 2 in 10 odstotkov vsake generacije;
- pri križanju smo za vsak parameter naključno izbrali eno izmed treh možnosti:
    - povprečje vrednosti obeh staršev;
    - vrednost prvega starša;
    - vrednost drugega starša.

Genetski algoritem smo ponavljali, dokler povprečni premik parametra med najboljšimi 5 primerki v generaciji ni dosegel definirane meje.

Za izvedbo genetskega algoritma nismo uporabili posebne knjižnice, saj gre za relativno preprost algoritem, ki se ga izvaja izključno v času priprave programske rešitve (določanja parametrov), zato optimizacija ni tako kritična.

4 PREDSTAVITEV IN VALIDACIJA REŠITVE

»Na Japonskem je bila skupina starejših gospodov, ki so se dobivali, da so si izmenjali novice in pili čaj. Bilo jim je v zabavo, do so izbirali dragocene vrste čaja in ustvarjali nove mešanice, ki so jih navduševale z izbranim okusom.

Ko je bil na vrsti najstarejši član skupine, da postreže drugim, je pripravil čaj zelo svečano; čajne lističe je vsipal iz zlate posode. Vsi so čaj izredno pohvalili in hoteli vedeti, s kakšnimi posebnimi razmerji je dosegel to odlično mešanico.

Stari mož se je smehljal in rekel: »Gospodje, čaj, ki se vam zdi tako okusen, je tisti, ki ga pijejo kmetje na mojem posestvu. Najboljše stvari v življenju niso niti drage niti jih ni težko najti.« « [62]

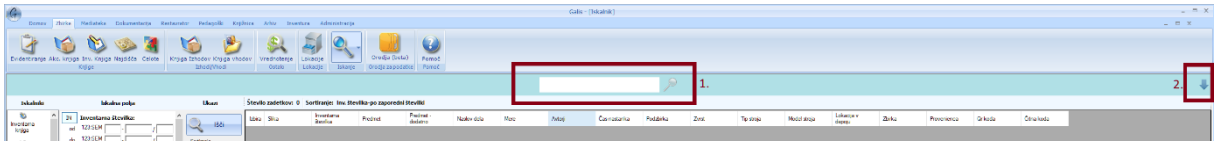
V okviru predhodnih poglavji smo predstavili motivacijo in teoretične osnove za izgradnjo dopolnitev programskega paketa Galis, ki so predmet te diplome, v tem poglavju pa se bomo posvetili pregledu izvedenih rešitev, predstavili nekaj zaslonskih slik in pogledali nekaj primerov razpoznav izvedenih z algoritmi razvitimi v okviru diplomskega dela.

4.1 Predstavitev rešitve

4.1.1 Polno besedilno iskanje

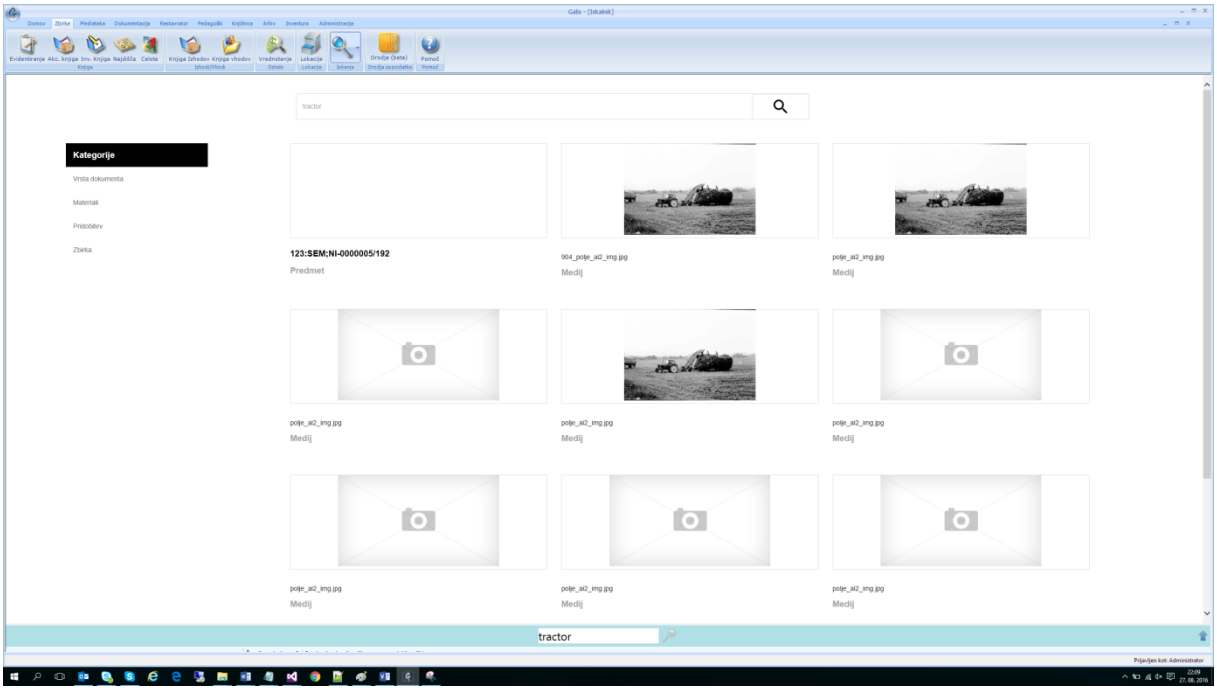
Funkcionalnost polnega besedilnega iskanja smo v Galis integrirali neposredno v obstoječi iskalnik z uporabo posebne iskalne vrstice na vrhu iskalnika, kot to prikazujemo na sliki 4.1. Do novega iskalnika lahko dostopamo na dva načina:

- z vnosom iskalne poizvedbe in pritiskom na tipko Enter v vrhnjo vrstico,
- s klikom na puščico na desni strani vrhnje vrstice.



Slika 4.1: Integracija iskalnika po polnem besedilu v Galis

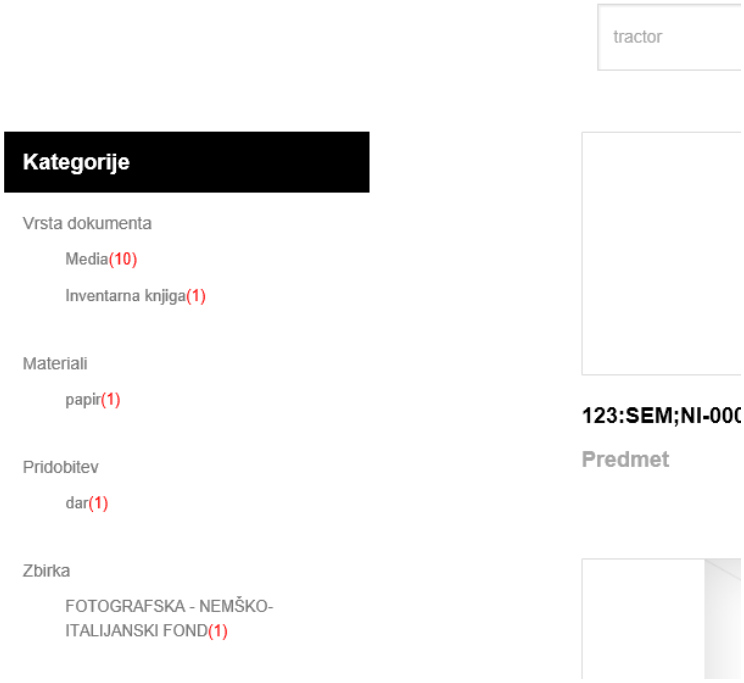
V obeh primerih se ob kliku razpre nov iskalnik po polnem besedilu, v primeru, da smo vnesli iskalno poizvedbo, dobimo iskalnik z že vneseno poizvedbo in rezultate, kot je razvidno na sliki 4.2.



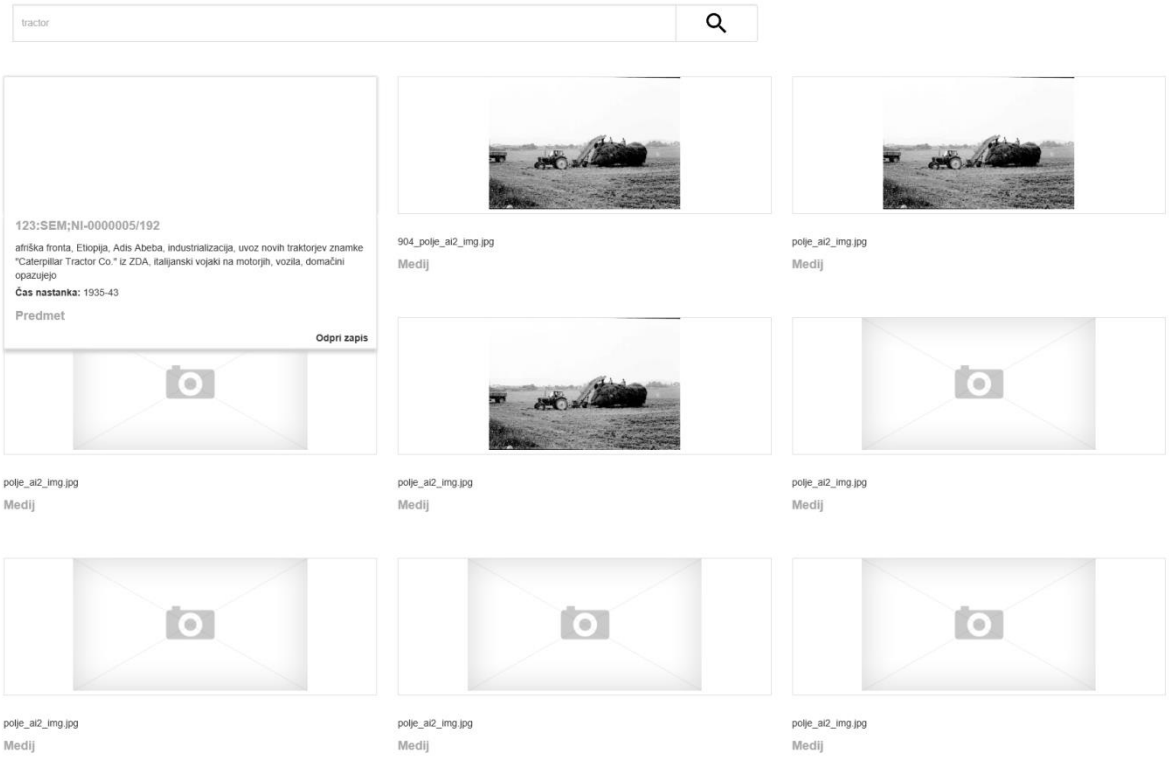
Slika 4.2: Iskalni rezultati za vnos "tractor" v testni bazi

Iskalni rezultati so ločeni na dva dela:

- na levi strani so izpisane kategorije predlaganih dodatnih filtriranj (slika 4.3),
- na desni so izpisani iskalni zadetki (slika 4.4).



Slika 4.3: Dodatna filtriranja iskalnih zadetkov



Slika 4.4: Prikaz iskalnih zadetkov, prvi zadetek je razširjen

V predhodnih poglavjih smo omenili tudi možnost kombiniranja polnega tekstovnega iskanja z različnimi filtri. Na sliki 4.5 je prikazan primer takega iskanja za iskalni pojem »čebele« in z aktiviranim dodatnim filtrom iskanja po avtorju »Edi Šelhaus«.

Slika 4.5: Iskanje z aktiviranim filtrom

4.1.2 Priprava predlogov vrednosti polj

Priprava predlogov vrednosti polj temelji na enaki tehnologiji in na istem kazalu za iskanje po polnem besedilu, kot iskalnik predstavljen v poglavju 4.1.1. Na sliki 4.6 je prikazan primer dodajanja novega vnosa v zbirko z imenom »FOTOGRAFSKA – EDI ŠELHAUS«. Ob kliku na gumb »Dodaj nov pametni predmet« nam Galis samodejno ponudi predloge za polja »klasifikacija«, »predmet« in »avtor«.

Slika 4.6: Predlagane vrednosti za nekatera polja glede na obstoječe vnose v bazi

4.1.3 Segmentacija in razpoznavanje slik

V okviru segmentacije in razpoznavanja slik smo v aplikacijo dodali samodejno analizo slike po tem, ko jo uporabnik doda. Primer samodejne razpoznave objekta je prikazan na sliki 4.7.

Slika 4.7: Samodejna zaznava objekta na sliki, ki ga razpoznamo kot "traktor"

4.2 Meritve rešitve

V okviru procesa validacije rešitve smo izmerili:

- povprečni čas čakanja na rezultate iskanja,
- povprečni čas čakanja na predloge vrednosti polj,
- pravilnost razpoznavanja slik.

Vse meritve so bile izvedene na naslednji strojni opremi:

- Intel Core i7-6600U @ 2.6 GHz,
- 16 GB RAM,
- Intel HD Graphics 520,
- 512 GB SSD disk.

Na računalnik je bil nameščen operacijski sistem Microsoft Windows 10 Enterprise, v času izvajanja je v ozadju tekla programska oprema Office 2016 in nekaj drugih aplikacij, da bi se čim bolj približali realnemu scenariju uporabe aplikacije. Strežnik in odjemalec sta bila za namene testiranja na istem računalniku.

4.2.1 Čas izvajanja iskanja

Izmerjene povprečne čase iskanja za različne scenarije iskanja prikazujemo v Tabeli 1

Tabela 1: Meritve časa izvajanja iskanja

Meritev	Povprečni čas [ms]
Čas prvega iskalnega zahtevka	34.586
Čas naslednjih iskalnih zahtevkov brez dodatnih filtrov	317
Čas naslednjih iskalnih zahtevkov z aktivnimi filtri	381

Meritve so bile dobljene tako, da smo 100 krat izvedli iskanje z naključno generirano iskalno poizvedbo iz vnaprej poznanih iskalnih besed, ki je vrnila vsaj 3 zadetke. V primeru iskanja s filtri smo naključno dodali 1, 2 ali 3 filtre iz nabora filtrov, ki jih je vrnila iskalna poizvedba brez filtra. Čas predstavlja razliko v času od trenutka sprožitve zahtevka do trenutka, ko je odjemalec prejel in obdelal rezultat in je zaokrožen na celo milisekundo.

V tabeli 1 vidimo, da je čas iskanja prvega iskalnega zadetka okrog 35 sekund, medtem ko se časi nadaljnjih iskanj gibljejo okrog 300 ms. Taki rezultati so povsem v skladu s pričakovanji, če se spomnimo, da je rešitev izvedena v okoljih .NET in da Lucene dodatno teče še znotraj javanskega virtualnega stroja. To pomeni, da moramo ob prvem zahtevku:

- prevesti vmesno kodo za iskalno storitev in jo zagnati v CLR;
- prevesti vmesno kodo za Lucene in jo zagnati s pomočjo IKVM.NET;
- naložiti iskalna kazala in »ogreti« iskalnik.

Vidimo tudi, da dodajanje filtrov ne poveča bistveno časa iskanja.

4.2.2 Čas izvajanja iskanja predlaganih vrednosti polj

Izmerjene povprečne čase pridobivanja podatkov za različne scenarije predlaganih vrednosti prikazujemo v tabeli 2.

Tabela 2: Meritve časa izvajanja iskanja predlaganih vrednosti

Meritev	Povprečni čas [ms]
Čas prvega zahtevka	33.915
Čas izvajanja iskanja predlogov naslednjih zahtevkov	344

Meritve so bile dobljene tako, da smo 15-krat izvedli zahtevek za pridobitev predlaganih vrednosti. Čas predstavlja razliko v času od trenutka sprožitve zahtevka do trenutka, ko je odjemalec prejel in obdelal rezultat in je zaokrožen na celo milisekundo.

V tabeli 1 vidimo, da je čas iskanja prvega iskalnega zadetka okrog 35 sekund, medtem ko se časi nadaljnjih iskanj gibljejo okrog 350 ms. Taki rezultati so povsem v skladu s pričakovanji, iz enakih razlogov, kot smo jih pojasnili že v poglavju 4.2.1.

4.2.3 Pravilnost razpoznanih segmentov

Za izračun pravilnosti razpoznanih segmentov smo združili izračun pravilnosti območja segmenta in objekta v danem segmentu na sliki. To pomeni, da smo segment šteli za pravilno razpoznan (*ang. true positive*), kadar se površina zaznanega segmenta med 90% in 110% prekriva z definiranim segmentom. Rezultat meritev prikazujemo v tabeli 3.

Tabela 3: Meritve pravilnosti razpoznave predmetov na slikah

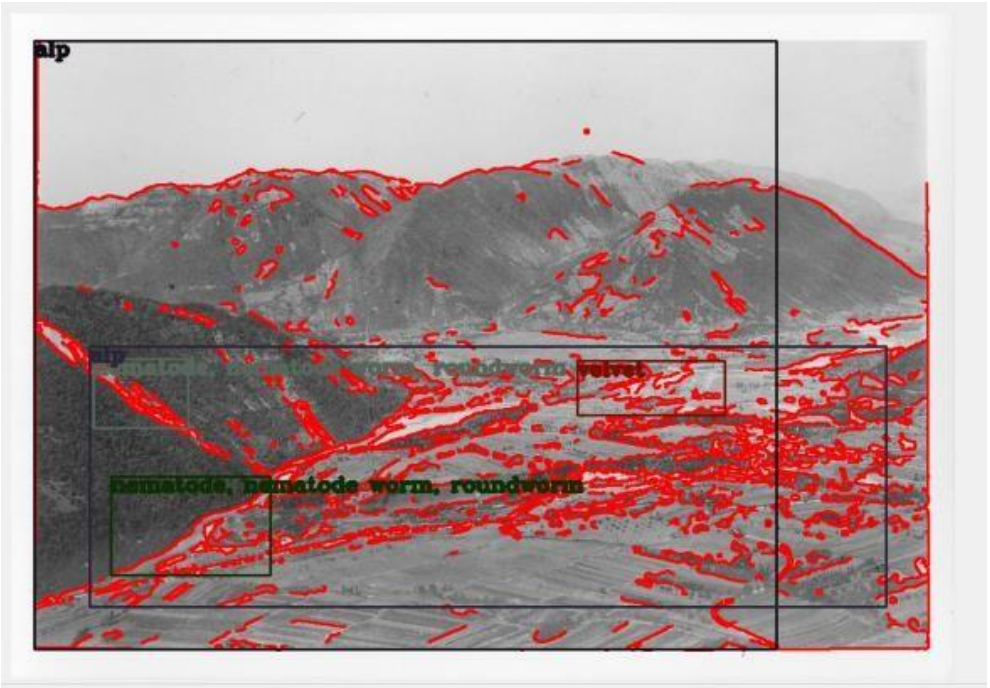
Rezultat	Število
Pravilen	42
Nepravilen	58
Skupaj	**100**

Kot vidimo iz rezultata smo pravilno zaznali 42 % segmentov, napačno pa 58 % segmentov, kar je pričakovano glede na to, da segmentacijski algoritmi, ki zmagujejo na tekmovanjih segmentacije, dosegajo nekje 56% zanesljivost (*ang. accuracy*) [37]; naš algoritem pa ima torej 42% zanesljivost.

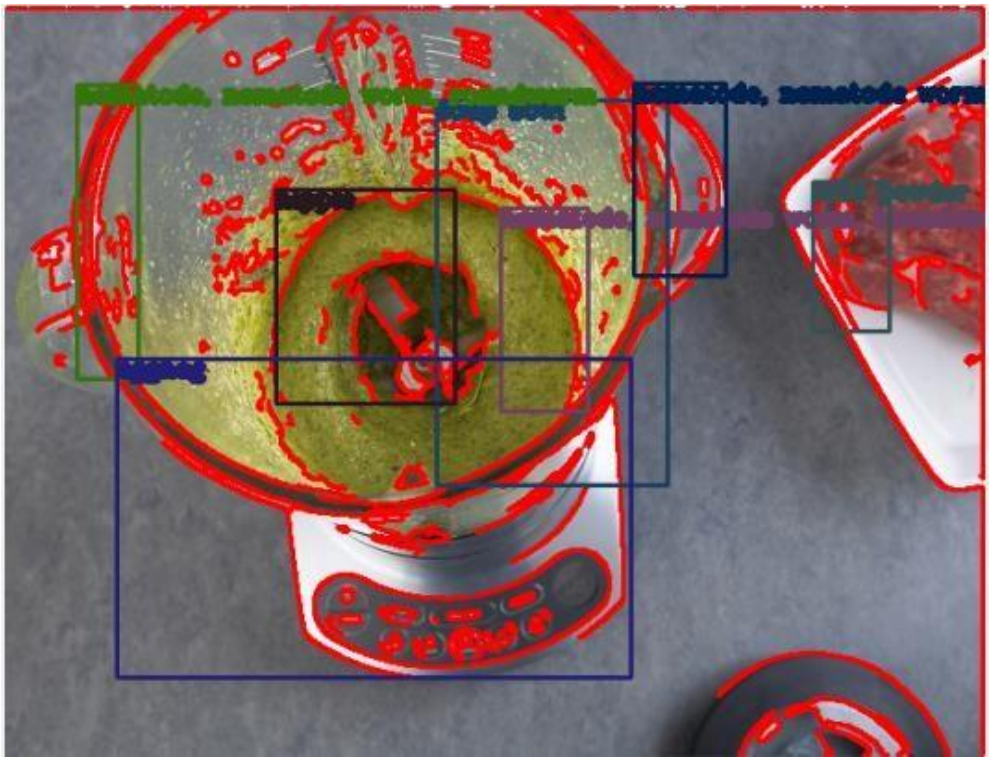
Kljub na prvi pogled slabim rezultatom je tak algoritem še zmeraj praktično uporaben, kar podrobneje pojasnjujemo v poglavju 5.

Nekaj primerov razpoznav za lažjo predstavo podajamo v slikah 4.8, 4.9 in 4.10, kjer prikazujemo primer dobre razpoznave, srednje dobre razpoznave in slabe razpoznave.

Slika 4.8: Dobra razpoznava - objekt zaznan in pravilno identificiran



Slika 4.9: Srednje dobra razpoznava - pravilno zaznane Alpe, zaznanih preveč objektov



Slika 4.10: Slaba razpoznava - zaradi pretirane segmentacije noben objekt zaznan pravilno

5 SKLEP

»V tovarni je bil čas opoldanskega odmora in neki delavec je žalostno odvil zavitek z malico. »Ah ne,« je rekel glasno, »že spet sendviči s sirom!«

Tako se je zgodilo še drugi in tretji in četrti dan. Potem je sodelavec, ki je slišal moževo godrnjanje pripomnil: »Če tako zelo sovražiš sendviče s sirom, zakaj ne naročiš ženi, naj ti jih pripravi s čim drugim?«

»Ker nisem poročen. Sendviče si delam sam.« « [63]

5.1 Diskusija

V okviru rezultatov smo predstavili meritve dveh pomembnih metrik za praktično uporabnost produkta razvitega v okviru diplomske naloge: časa izvajanja in pravilnosti razpoznave.

Iz rezultatov je razvidno, da so časi iskanja dovolj dobri, da jih lahko brez težav uporabimo v komercialni rešitvi in omogočajo učinkovito in hitro delo z aplikacijo ter izboljšujejo uporabniško izkušnjo, ki so jo uporabniki Galisa imeli pred uvedbo novega modula. Zaključimo lahko torej, da sta funkcionalnosti hitrega iskanja po polnem besedilu in predlogov vrednosti polj pri novih vnosih primerni za neposredno implementacijo v Galis.

Nekaj več razmisleka terjajo predstavljeni rezultati segmentacije, kjer je stopnja razpoznave 42% na prvi pogled dokaj slaba. Pri tem se spomnimo, kako smo definirali pravilno razpoznavo: omejujoči pravokotnik zaznanega objekta se mora precej dobro prilegati omejujočemu pravokotniku, ki smo ga definirali kot pravilnega, hkrati pa mora naš algoritem tudi pravilno identificirati objekt znotraj pravokotnika. Hitro vidimo, da je ta pogoj precej strog in da za praktično uporabo v veliko primerih zadostuje, da pravilno razpoznamo že omejujoč pravokotnik in uporabniku ponudimo več možnosti, kaj se znotraj pravokotnika nahaja. Predvidevamo, da bomo v tem primeru dobili precej boljše rezultate, kar bomo v prihodnosti preverili.

5.2 Zaključki

Poudariti je pomembno tudi, da je bil cilj diplomske naloge razviti prvo iteracijo takega produkta, ki uporablja najnovejše (*state of the art*) tehnologije s področja računalniškega vida, ki jih bomo znotraj razvojno raziskovalnega oddelka Semantike v prihodnosti dodatno nadgradili, pri čemer pričakujemo precejšnje izboljšanje stopnje razpoznavanja. Tako bomo v prihodnosti algoritem za razpoznavo predmetov dopolnili z ansambelskimi metodami, ki jih bomo dodatno kalibrirali s pomočjo genetskega algoritma, segmentacijski algoritem pa

bomo dopolnili, da bo zraven preprostih metod segmentacije, kot je sledenje robov, uporabljal še metode strojnega učenja kot so konvolucijske nevronske mreže.

Na podlagi vsega, kar smo navedli v sklepu, lahko zaključimo, da je diplomsko delo svoj cilj doseglo. V njegovem okviru smo izdelali delujoče rešitve za polno besedilno iskanje, predloge vrednosti polj in segmentacijo slik ter na podlagi razvitih algoritmov in rešitev dopolnili obstoječ komercialni produkt Galis.

6 VIRI

[1] A. Bloch, „Murphyjev zakon ter ostali razlogi, da stvari gredo narobe,“ 1991.

[2] D. Adams, The Hitchhiker's Guide to the Galaxy, New York: Ballantine books, 2009.

[3] Evropska skupnost, „Kratki vodnik po Evropski uniji,“ ES, 2016. [Elektronski]. Available: http://www.europarl.europa.eu/atyourservice/sl/displayFtu.html?ftuId=FTU_5.13.1.html. [Poskus dostopa 18 08 2016].

[4] Evropska komisija, „Strategic framework - European Agenda for Culture,“ [Elektronski]. Available: http://ec.europa.eu/culture/policy/strategic-framework/index_en.htm. [Poskus dostopa 20 8 2016].

[5] Intel, „Intel® SDK for OpenCL™ Applications,“ [Elektronski]. Available: https://software.intel.com/en-us/intel-opencl. [Poskus dostopa 01 08 2016].

[6] Nvidia, „About CUDA,“ [Elektronski]. Available: https://developer.nvidia.com/about-cuda. [Poskus dostopa 02 08 2016].

[7] Wikimedia, „Warren Bennis,“ [Elektronski]. Available: https://en.wikiquote.org/wiki/Warren_Bennis. [Poskus dostopa 22 08 2016].

[8] Semantika d.o.o., „Kulturna dediščina,“ Semantika d.o.o., 2016. [Elektronski]. Available: http://www.semantika.si/Projekti/Kulturna-dedi%C5%A1%C4%8Dina. [Poskus dostopa 6 08 2016].

[9] Semantika d.o.o., *Statistika uporabe programske opreme Galis 2016,* Maribor, 2016.

[10] Semantika d.o.o., *Galis - muzejski dokumentacijski sistem,* Maribor: Semantika d.o.o., 2010.

[11] Semanitka d.o.o., „Opis programskega paketa Galis,“ Maribor, 2016.

[12] INTERNATIONAL COUNCIL ON ARCHIVES, „ISAD(G),“ [Elektronski]. Available: http://www.icacds.org.uk/eng/ISAD(G).pdf. [Poskus dostopa 27 8 2016].

[13] Wikipedia, „.NET Framework,“ [Elektronski]. Available: https://en.wikipedia.org/wiki/.NET_Framework. [Poskus dostopa 6 8 2016].

[14] Microsoft, „Common Language Runtime (CLR),“ [Elektronski]. Available: https://msdn.microsoft.com/en-us/library/8bs2ecf4(v=vs.110).aspx. [Poskus dostopa 25 8 2016].

[15] Microsoft, „An Overview of Managed/Unmanaged Code Interoperability,“ [Elektronski]. Available: https://msdn.microsoft.com/en-us/library/ms973872.aspx. [Poskus dostopa 25 8 2016].

[16] Microsoft, „SELECT (Transact-SQL),“ [Elektronski]. Available: https://msdn.microsoft.com/en-us/library/ms189499.aspx. [Poskus dostopa 24 8 2016].

[17] Microsoft, „Understanding Indexes,“ [Elektronski]. [Poskus dostopa 18 8 2016].

[18] Microsoft, „Types of Indexes,“ [Elektronski]. [Poskus dostopa 18 8 2016].

[19] Microsoft, „Index Basics,“ [Elektronski]. [Poskus dostopa 18 8 2016].

[20] Wikipedia, „Full text search,“ [Elektronski]. Available: https://en.wikipedia.org/wiki/Full_text_search. [Poskus dostopa 27 7 2016].

[21] O. Gospodnetic in E. Hatcher, Lucene in action, Manning, 2005.

[22] Wikimedia, „Fulltext search engines,“ [Elektronski]. Available: https://www.mediawiki.org/wiki/Fulltext_search_engines. [Poskus dostopa 5 7].

[23] „Apache Lucene Core,“ [Elektronski]. Available: http://lucene.apache.org/core/. [Poskus dostopa 1 8 2016].

[24] „Solr Features,“ [Elektronski]. Available: http://lucene.apache.org/solr/features.html. [Poskus dostopa 1 8 2016].

[25] „Elasticsearch,“ [Elektronski]. Available: https://www.elastic.co/products/elasticsearch. [Poskus dostopa 1 8 2016].

[26] „FlexLucene,“ [Elektronski]. Available: https://www.nuget.org/packages/FlexLucene/.

[27] „Introduction,“ [Elektronski]. Available: https://www.ikvm.net/#Introduction.

[28] Wikipedia, „Image segmentation,“ [Elektronski]. Available: https://en.wikipedia.org/wiki/Image_segmentation#Watershed_transformation. [Poskus dostopa 20 7 2016].

[29] Y.-J. e. Zhang, „Advances in image and video segmentation,“ 2006.

[30] N. R. Pal in S. K. Pal., „A review on image segmentation techniques.,“ *Pattern recognition 26.9,* 1993.

[31] R. Unnikrishnan, C. Pantofaru in M. Herbert, „Toward objective evaluation of image segmentation algorithms.,“ *IEEE transactions on pattern analysis and machine intelligence,* 2007.

[32] H. e. a. Müller, „Performance evaluation in content-based image retrieval: overview and proposals,“ *Pattern Recognition Letters 22.5,* 2001.

[33] C. e. a. Szegedy, „Going deeper with convolutions,“ v *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[34] J. Long, E. Shelhamer in T. Darrel, „Fully Convolutional Networks for Semantic Segmentation,“ v *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[35] F. N. e. a. Iandola, „FireCaffe: near-linear acceleration of deep neural network training on compute clusters.,“ *arXiv preprint arXiv:1511.00175,* 2015.

[36] K. Simonyan in A. Zisserman, „Very deep convolutional networks for large-scale image recognition.,“ *arXiv preprint arXiv:1409.1556,* 2014.

[37] P. e. a. Sermanet, „Overfeat: Integrated recognition, localization and detection using convolutional networks.,“ *arXiv preprint arXiv:1312.6229,* 2013.

[38] A. H. Wright, „Genetic algorithms for real parameter optimization,“ *Foundations of genetic algorithms 1,* 1991.

[39] T. Bäck in H.-P. Schwefel, „An overview of evolutionary algorithms for parameter optimization,“ *Evolutionary computation 1.1,* 1993.

[40] Wikipedia, „OpenCV,“ [Elektronski]. Available: https://en.wikipedia.org/wiki/OpenCV. [Poskus dostopa 13 8 2016].

[41] OpenCV, „OpenCV license,“ [Elektronski]. Available: http://opencv.org/license.html. [Poskus dostopa 27 8 2016].

[42] Wikipedia, „Comparison of deep learning software,“ Wikipedia, [Elektronski]. Available: https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software. [Poskus dostopa 22 7 2016].

[43] Berkley Vision Laboratory, „Why Caffe?,“ [Elektronski]. Available: http://caffe.berkeleyvision.org/. [Poskus dostopa 10 7 2016].

[44] Caffe, „Model Zoo,“ Berkley Vision Lab, [Elektronski]. Available: https://github.com/BVLC/caffe/wiki/Model-Zoo. [Poskus dostopa 22 07 2016].

[45] About.com, „What Are Clarke's Laws?,“ [Elektronski]. Available: http://physics.about.com/od/physics101thebasics/f/ClarkesLaws.htm. [Poskus dostopa 20 8 2016].

[46] Microsoft, „ASP.NET Overview,“ Microsoft, [Elektronski]. Available: https://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx. [Poskus dostopa 1 8 2016].

[47] S. Hunter, „.NET Core Roadmap,“ Microsoft, [Elektronski]. Available: https://blogs.msdn.microsoft.com/dotnet/2016/07/15/net-core-roadmap/. [Poskus dostopa 25 7 2016].

[48] Apache Lucene, „Lucene 6.1.0 Documentation,“ Apache Lucene, [Elektronski]. Available: http://lucene.apache.org/core/6_1_0/. [Poskus dostopa 22 7 2016].

[49] Apache Lucene, „Lucene 6.1.0 core API,“ [Elektronski]. Available: http://lucene.apache.org/core/6_1_0/core/index.html. [Poskus dostopa 5 08 2016].

[50] Apache Lucene, „Lucene60Codec,“ Lucene, [Elektronski]. Available: http://lucene.apache.org/core/6_1_0/core/index.html. [Poskus dostopa 10 8 2016].

[51] Apache Lucene, „Class TFIDFSimilarity,“ [Elektronski]. Available: https://lucene.apache.org/core/6_1_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html. [Poskus dostopa 7 8 2016].

[52] C. D. Manning, P. Raghavan in H. Schütze, „Queries as vectors,“ [Elektronski]. Available: http://nlp.stanford.edu/IR-book/html/htmledition/queries-as-vectors-1.html.

[53] K. Rithwik in K. N. Chaudhury, „A Simple Yet Effective Improvement to the Billateral Filter for Image Denoising,“ *arXiv:1505.06578v1,* 2015.

[54] N. Otsu, „A threshold selection method from gray-level histograms,“ *IEEE Trans. Sys., Man., Cyber.,* 1979.

[55] J. Canny, „A computational approach to edge detection,“ *IEEE Transactions on pattern analysis and machine intelligence 6,* 1986.

[56] S. Suzuki in K. Abe, „Topological Structural Analysis of Digitized Binary Images by Border Following,“ *CVGIP 30 1,* 1985.

[57] D. Luebke, „A survey of polygonal simplification algorithms.,“ *Dept. Computer Science, University of North Carolina, Chapel Hill, Tech. Rep. TR97-045,* 1997.

[58] F. Chin in C. A. Wabg, „Optimal algorithms for the intersection and the minimum distance problems between planar polygons,“ *IEEE Transactions on Computers 32.12,* 1983.

[59] Berkely Vision Laboratory, „Berkley trained Model,“ [Elektronski]. Available: https://github.com/BVLC/caffe/wiki/Model-Zoo#berkeley-trained-models. [Poskus dostopa 25 07 2016].

[60] A. D. Mello, „Navaden čaj je najboljši,“ v *Žabja molitev 2*, Ljubljana, Župnijski zavod Lj.-Dramlje, 2009.

[61] A. D. Mello, „Kdo dela sendviče,“ v *Žabja molitev 2*, Ljubljana, Župnijski zavod Lj.-Dravlje, 2009.

[62] N. Senthilkumaran in R. Rajesh, „Edge detection techniques for image segmentation–a survey of soft computing approaches,“ *International journal of recent trends in engineering,* Izv. 1.2, 2009.