SVEUČILIŠTE U DUBROVNIKU
ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO
STUDIJ POSLOVNOG RAČUNARSTVA

Marko Cetinić

DIPLOMSKI RAD

PRIMJENA OCR I TTS TEHNOLOGIJA U KONTEKSTU PROŠIRENE STVARNOSTI

Dubrovnik, Svibanj 2012

SVEUČILIŠTE U DUBROVNIKU
ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO
STUDIJ POSLOVNOG RAČUNARSTVA

DIPLOMSKI RAD

PRIMJENA OCR I TTS TEHNOLOGIJA U KONTEKSTU PROŠIRENE STVARNOSTI

Mentor:
dr. sc. Mario Miličević

Diplomant:
Marko Cetinić

Dubrovnik, Svibanj 2012

REPUBLIKA HRVATSKA
SVEUČILIŠTE U DUBROVNIKU
ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO
STUDIJ: Studij Poslovnog Računarstva
Sveučilišni diplomski studij

Ur. broj: Dubrovnik, Svibanj 2012

Kolegij: <naziv kolegija>
Mentor: dr. sc. Mario Miličević

D I P L O M S K I Z A D A T A K

Kandidat: **Marko Cetinić**

Broj indeksa: **<broj indeksa>**

Naslov zadatka: **PRIMJENA OCR I TTS TEHNOLOGIJA U KONTEKSTU PROŠIRENE STVARNOSTI**

Opis zadatka: Potrebno je: dati pregled razvoja i primjene tehnoloških rješenja iz područja proširene stvarnosti (Augmented Reality). Dati pregled razvoja OCR (Optical Character Recognition) i TTS (Text to Speech) tehnologija i mogućnosti njihove primjene u kontekstu proširene stvarnosti. Projektirati i realizirati mobilnu aplikaciju koja će primijeniti navedene tehnologije. Analizirati moguća područja primjene realiziranog programskog rješenja.

Zadatak je uručen kandidatu <datum>
Rok za predaju diplomskog rada je do: 17. svibnja

Mentor:
dr. sc. Mario Miličević

Pročelnik Odjela: */ili Pročelnica!/*
<titula> <ime i prezime >

Sadržaj

Uvod

Stvarnost ili **Zbilja** je pojam koji označava ono što stvarno postoji[1].

Šira definicija stvarnosti uključuje sve što je sad i što je nekad bilo, unatoč tomu dali se može percipirati ili pojmiti. Filozofi i matematičari kao Aristotel, Platon, Frege, Wittgenstein, Russell itd. smatraju da postoji razlika između stvarnosti koju smo u stanju racionalno pojmiti i fizičke "prave" stvarnosti koju nismo u stanju percipirati.

Iako većina ljudi slično percipira stvarnost, njezino poimanje je ograničeno našim čulima; vid, sluh, njuh i opip. Ograničeno je još biokemijom naših tijela i socijalno-geografskim okruženjem. Stvarnost kao takva u višoj ili manjoj mjeri varira od osobe do osobe, od društva do društva.

Stvarnost je često u kontrastu s imaginacijom, iluzijama, apstrakcijom i fikcijom. **Istina** *(true)* se odnosi na ono što je stvarno, a **Laž** *(flase)* na ono što nije.

Proširena Stvarnost *(Augmented Reality – AR)* je trenutni, direktni ili indirektni pogled na fizički okoliš čiji su elementi poboljšani pomoću računalno generiranih senzorskih doprinosa *(sensory input)* – kao zvuk, video, grafika ili GPS *(Global Positioning System)* podatci[2]. Proširena stvarnost se odnosi na širi pojam zvan **Modificirana Stvarnost** *(Mediated Reality)*[2], kad stvarnost ne mora nužno biti poboljšana. Svrha ove tehnologije je povećanje individualne percepcije stvarnosti. Nadskup Proširenoj stvarnosti je **Virtualna Stvarnost** *(Virtual Reality)* koja u potpunosti zamjenjuje stvarni svijet s simuliranim.

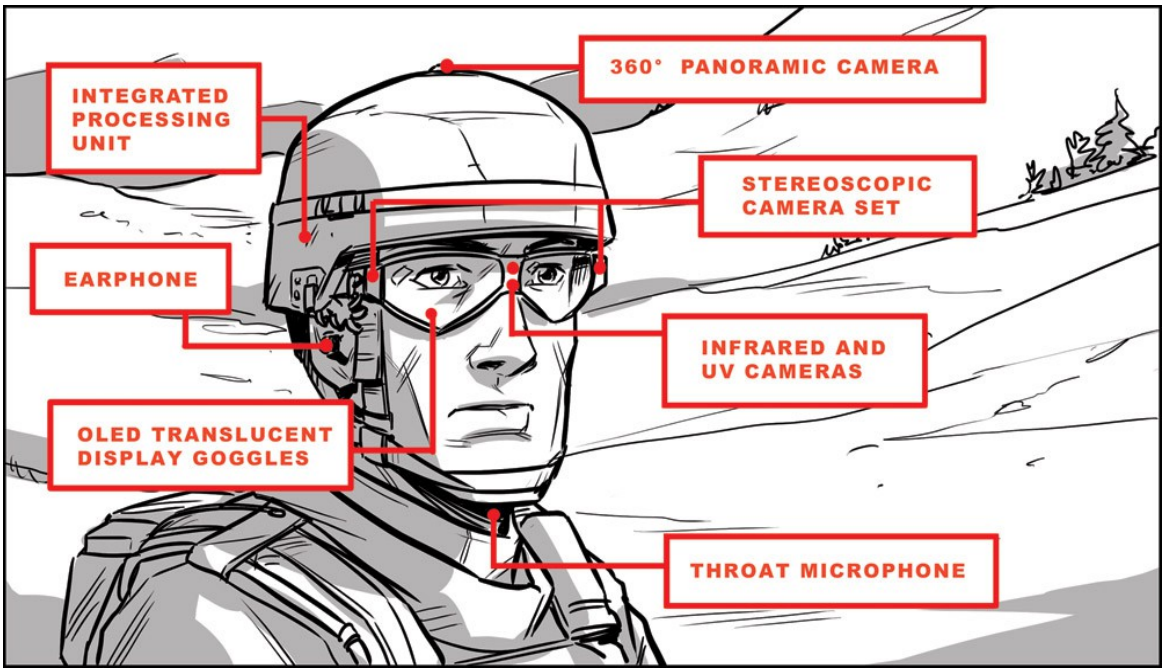
Proširenje je u stvarnome vremenu *(real-time),* te u kontekstu okruženja, kao na primjer rezultat/statistika na TV ekranu za vrijeme utakmice. Kod naprednijih **AR** tehnologija, informacije o okruženju postaju interaktivne te se može s njima manipulirati. Umjetne informacije o okolišu i njegovom okruženju se mogu prikazati u kombinaciji s "pravom" stvarnošću. Iako se pojam proširene stvarnosti javlja još 1957 u kinematografiji smatra se da je termin *Proširena Stvarnost* stvorio 1990 Thomas Caudell dok je radio za Boeing[21].

Kao jedan od načina proširenja stvarnosti smatra se primjena računalno generiranih vizualnih elemenata unutar video struja *(streaming)*. AR tehnologija uključuje kacige s vizualnim pomagalima *(Head up Display – HUD)*, te stvara kontrolirani okoliš. Osim HUD – a kao vizualno pomagalo može poslužiti bilo koji uređaj s mogućnošću snimanja – na primjer kamera mobitela.

Brzina razvoja Proširene stvarnosti se povećala otkako novi pametni telefoni *(smartphones)* kao iPhone 3GS, te uređaji koji koriste Android operacijski sustav dolaze opremljeni s prijeko potrebnim komponentama za AR. Potrebne komponente su: GPS, kompas i kamera[21].

Tehnologija proširene stvarnosti, iako u procesu ubrzanog razvoja, još je relativno mlada, te se kao takva teško implementira u mobilinim uređajima. Nasuprot tome, velike korporacije u auto industriji kao BMW su iskazale veliki interes za nju, pogotovo kad se koristi u proizvodnji, te u kombinaciji s GPS sustavima. Zbog navedenog, predviđa se daljnji razvoj ove tehnologije[5].

Osim korporacija u privatnom vlasništvu, državne ustanove kao vojska SAD-a također razvijaju svoje inačice ove tehnologije. IT tvrtka u Chicagu zvana **Tanagram Partners**[6] trenutno je u procesu razvoja vojnog programa koji bi mogao, ako se pokaže uspješnim, promijeniti način ratovanja. Navedeni program bi radio kao HUD koji prima podatke koje vojnik vidi i šalje poslužitelju *(server)* u bazi. Poslužitelj analizira dobivene podatke, te ih vraća da bi se prikazale na istom HUD – u. Time se vojniku šalju taktički podatci o okolišu u kojem se nalazi, te bi se teoretski mogao i stvoriti diskriminator između civila i neprijateljskih vojnika. Dolje navedena fotografija prikazuje jedan takav HUD.



Slika 1. HUD

Nasuprot vojnih svrha, AR tehnologija koristi se se u polju medicine već 10 godina. Vizualizacija unutarnjih tjelesnih struktura može biti veoma moćan alat pri edukaciji i treningu studenata medicine. Osim navedenih prednosti postoji još primjena AR tehnologije kod Atroskopskih (keyhole) metoda[21] . To je minimalno invazivni operativni zahvat kojim se kroz male rezove, veličine do 1cm upotrebom optičkih instrumenata, sustava leća (koji su povezani s kamerom) i posebno dizajniranih malih instrumenata, cijeli tok operacije prati na monitoru. Slika iz zgloba uvećana je oko trideset puta. Primjenom AR – a, u mogućoj kombinaciji s rendgenom, može se uvelike olakšati vizualizacija tih istih struktura.

Slika 2. Vizualizacija operiranog zgloba

Uz gore navedene vojne i medicinske svrhe, AR ima veliki, većinom neiskorišten potencijal u domeni igara i mobilnih aplikacija. S jačanjem hardvera mobitela javljaju se sve bolji uvjeti za implementaciju navedene tehnologije, te su mnoge od razvojnih kuća već razvile prve **mobilne aplikacije** *(mobile apps*) koje sadrže elemente AR – a[23]. Dobar primjer uspješne primjene AR – a bila bi iPhone aplikacija *SnapShop Showroom*[7] koja omogućuje postavljanje virtualnog namještaja u praznu prostoriju. Fotografirira se željeni komad namještaja, te se istog nakon računalne obrade može prikazati unutar prostorije koja se snima. Uz pravilnu primjenu AR tehnologije lako bi se mogla napisati aplikacija koja simulira virtualni "lov na blago" te mnoge druge.

Cilj ovog rada je prikaz AR – a u kontekstu s **Tekst u Zvuk** *(Text to Speech – TTS)* i **Optičko prepoznavanje znakova** *(Optical Character Recognition – OCR).*

- OCR je sistem u stanju kao ulaz primiti sliku što sadrži tekstualne simbole, te kao izlaz vratiti tekstualnu datoteku koja sadrži tekst što odgovara tim simbolima[8].

- TTS kao ulaz prima tekstualnu datoteku, te kao izlaz vraća zvučnu datoteku koja sadržava zvuk što odgovara sadržaju tekstualne datoteke.

Nekadašnja implementacija ovih tehnologija je uključivala korištenje skenera i računala. S razvojem mobilne tehnologije otvaraju se nove mogućnosti i primjene.

Primjena OCR – a u kombinaciji s AR tehnologijom dolazi do značaja tek u zadnje vrijeme, iako i jedna i druga postoje već duže vremena. OCR tehnologija se koristi pri prepoznavanju tablica automobila u regulaciji prometa[9], pa se kao takva može smatrati određenom razinom proširene stvarnosti. Razlika između tradicionalne i ove AR primjene jest da se ne odvija u realnome vremenu i da ne mora biti interaktivna.

TTS u kombinaciji s AR, je sljedeća "velika stvar" u budućim službenim tehnologijama (*mainstream*). Sposobnost da se fotografija riječi pretvori u zvuk ima neograničen broj mogućih uporaba, ali se najviše očekuju rješenja za osobe s vizualnim problemima i u sistemu obrazovanja. Pisane greške bi mogle biti detektirane i objašnjene onog trenutka kad se dogode[10]. Učenje jezika bi uvelike bilo

pojedostavljeno, kao i praćenje nekog kompleksnog kuharskog recepta.

Kombinacija navedenih tehnologija objedinjena na mobilnoj platformi predstavlja snažnu kombinaciju s još neistraženim potencijalom. Turist koji se nalazi u nepoznatoj državi i ne govori lokalni jezik ili je pismo na drugačijem alfabetu (ćirilični, arapski itd..) može prevesti i čuti izgovor natpisa na prometnim znakovima. Potrebno je samo fotografirati znak (kao dolje naveden znak Stop), obraditi ga i poslušati rezultat obrade.

Slika 3. Znak stop

Aplikacija mora biti u stanju fotografirati površinu koja sadrži neki tekst i nakon obrade vratiti izrečeni zvuk tog teksta.

1 Povijesni pregled korištenih tehnologija

1.1 Povijest tehnologije AR – a

Termin "Proširena Stvarnost" postoji od 1990, ali to nije vrijeme kad je nastao. Onog trenutka kad je tehnološki razvoj omogućio čovjeku da prikaže okoliš kroz određeni video terminal, te ga opskrbi s dodatnim informacijama, AR je nastala. Jedino je još nedostajao odgovarajući naziv[2].

Od 1957. Morton Helig je počeo izgradnju uređaja zvanog **Sensorama**. Svrha mu je bila da pri kinematografskoj projekciji uključi i ostala osjetila osim vida i sluha. Sensorama bi puhao vjetar, vibrirao sjedalo i projektirao 3D okoliš, dizajniran pomoću tri kamere. Uređaj nije doživio komercijalni uspjeh zbog skupog načina izrade filmova, ali se smatra prvim zabilježenim pokušajem AR – a[2].

Godine 1966. Profesor Ivan Sutherland, s Odjela Elektronike na Sveučilištu Harvardu *(Harvard Office of Technology Development)*, izumio je prvi model uređaja potrebnog u AR – u i VR – u, HMD *(head-mounted display)*. Tadašnji primitivni HMD je bio prevelik i pretežak da ga se nosi na glavi, pa je zbog toga visio sa stropa. Bez obzira na njegov primitivni dizajn bio je prvi realni pokušaj primjene AR tehnologije u obliku kakav danas poznajemo[2].

Fraza "Proširena stvarnost" se oblikovala 1990. od strane Profesora Toma Caudella dok je radio za *Boeing računalne servise – adaptivni neuralni sistemi – Istraživanje i Razvoj odjel u Seattlu (Boeing's Computer Services' Adaptive Neural Systems Research and Development project in Seattle)*. U pokušaju da pojednostavni proces proizvodnje i inženjeringa, primjenjiivao je tehnologiju virtualne stvarnosti, te napokon stvorio kompleksni program koji prikazuje poziciju određenih komponenti (kabela). Time je eliminirao potrebu za suvišnim upitima mehaničara kad bi trebali pratiti nedovoljno detaljna upustva[2].

Godine 1992. L.B. Rosenberg razvija prvi u potpunosti funkcionalni AR sistem za Američko Ratno Zrakoplovstvo *(US Air Force)*. U isto vrijeme druga grupa koja se

sastoji od Stevena Feinera, Blaira MacIntyrea i Doreea Seligmanna, danas vodećih imena u polju AR – a, izdaje znanstveni rad na temu **Proširana Stvarnost kao pomoć pri radu** *(Knowledge-based Augmented Reality for Maintenance Assistance – KARMA)*[12]. Rad je veoma poznat u AR zajednici, te je dobio dobre kritike od strane znanstvene zajednice[2].

Do 1999. Proširena Stvarnost ostaje zanemareno polje računalne znanosti. To se mijenja iste godine kad Hirokazu Kato iz Nara Instituta Znanosti i Tehnologije *(Nara Institute of Science and Technology)* izdaje **ARToolKit**. Prvi put korisnicima je omogućeno praćenje okoliša pomoću kamere i proširivanje istog s virtualnim grafičkim elementima. Iako pametni telefoni tada još nisu izumljeni, bilo je moguće koristiti jednostavnu kameru da bi se AR ostvario. Skoro sve AR Flash aplikacije na pretraživačima su bazirane na ARToolKit – u[11].

Godinama kasnije – 2008, prva AR aplikacija dolazi na pametne telefone. Izdavačka kuća **Mobilizy** je među prvima koji su donijeli AR na operacijski sustav Android, te omogućili korisnicama, da pomoću kamere mobitela vide dodatne podatke snimljenih okruženja (ulica, zgrada, brda itd..)[3].

Gledajući realno, aplikacije bazirane na AR – u su još primitivne i razvojno u ranoj fazi, ali se može sa sigurnošću predvidjeti daljnji razvoj ove tehnologije[3].

1.2 Povijest tehnologije OCR-a

Rani sistemi prepoznavanja simbola su nastali početkom 20. stoljeća iz dva razloga:

- razvoj uređaja za pomoć slijepim
- širenje telegrafije[8].

Godine 1914. Emanuel Goldberg razvija uređaj koji čita simbole i pretvara ih u telegrafski kod. U isto vrijeme, Edmund Fourier d'Albe razvija **Optophone** – skener koji pri pomicanju preko ispisane stranice, proizvodi tonove koji odgovaraju tim istim simbolima.

Goldberg nastavlja razvijati OCR tehnologiju s preporukom da bi trebalo

fotografirati podatke, nakon čega koristeći foto – ćelije uspoređivati fotografiju s predloškom koji sadrži identifikacijski uzorak.

Godine 1929. Gustav Tauschek u Njemačkoj patentira sličnu ideju.

Godine 1933. Paul W. Handel pantentira svoju OCR tehnologiju, ali za razliku od Tauscheka patent dobiva u SAD – u (U.S. Patent 1,915,993).

Godine 1935. Tauschek također dobavlja patent u SAD – u (U.S. Patent 2,026,329).

Godine 1949. RCA inženjeri stvaraju prvi primitivni računalni OCR u svrhu pomaganja slijepim američkim ratnim veteranima, ali umjesto samog konvertiranja ispisanih simbola u strojni jezik, njihovi uređaji su ih još i izgovarali. Projekt nikad nije dovršen zbog prevelikih troškova izrade.

Godine 1950. David H. Shepard, kriptoanalitičar pod AFSA *(Armed Forces Security Agency)* u SAD-u, zbog problema konverzije ispisanih poruka u strojni jezik stvara uređaj zvan "**Gismo**", koji kasnije i pantentira (U.S. Patent 2,663,758). Gismo je mogao čitati 23 slova engleskog alfabeta, razumio je Morsov kod i muzičke notacije. Shepard kasnije osniva *Intelligent Machines Research Corporation (IMR)* koji uskoro razvija prvi komercijalni OCR sistem.

Godine 1955, taj isti sistem je instaliran u časopisu *Reader's Digest*, koji je koristio OCR da unese izvješća o prodaji u računalo. Konvertirao je ispisana izvješća u bušene kartice *(punch card)*, čime je pomagao u isporuci 15 – 20 milijuna knjiga godišnje. Još jedan sistem je prodan *Standard Oil Company* kojem je čitao kreditna izvješća. Drugi sistemi su prodani *Ohio Bell Telephone Company* i *US Air Force* za svrhu čitanja i slanja pisanih poruka. IBM je kasnije kupio licence za Shepardov OCR patent.

Oko 1965, Reader's Digest i RCA ostvaruju suradnju da izgrade OCR čitač dokumenata sa svrhom digitaliziranja serijskih brojeva reklamacijskih kupona. OCR čitač je radio u kombinaciji s RCA 301 računalom, te je bio u stanju pročitati oko 1500 dokumenata po minuti.

Poštanska služba SAD – a *(The United States Postal Service)* koristi OCR

uređaje pri ažuriranju pošte još od 1965. Prvi OCR u Europi se također koristio za ažuriranje pošte ali ovaj put u Velikoj Britaniji *(British General Post Office – GPO).* Godine 1965. sistem plaćanja u VB je revolucioniran korištenjem OCR tehnologije. Kanadska pošta koristi OCR od 1971. OCR sistem čita ime i adresu u prvom ažurirajućem centru, te ispisuje bar kod *(bar code).*

1.3 Povijest tehnologije TTS – a

Godine 1779. Danski Znanstvenik Christian Kratzenstein, radeći za Rusku Akademiju Znanosti, izgradio je model ljudskog trakta koji je u stanju stvoriti pet samoglasnika (a, e, i, o, u)[10].

Nakon toga, god. 1791. Wolfgang von Kempelen iz Pressburga (Mađarska) je opisao mijeh koji simulira glas.

Godine 1837. Charles Wheatstone stvara "**Stroj za govor**" *(Talking Machine)* prema dizajnu Wolfgang von Kempelena.

Godine 1857. M. Faber stvara "**Euphoniu**". Isti taj uređaj je 1923. preuredio Paget.

Godine 1930, Bell Laboratorij (Bell Labs) razvija **VOCODER**, elektronički uređaj s kojim se upravlja tipkovnicom, te se smatra prvim kojeg je bilo moguće u potpunosti razumjeti. Homer Dudley prerađuje ovaj uređaj u **VODER**, koji predstavlja 1939. godine na svjetskom sajmu u New York – u *(New York World's Fair)*.

Kasne 1940, te rane 1950. – Dr. Franklin S. Cooper i njegovi suradnici u Haskins Laboratoriju izgradili su **Igrač Uzoraka** *(The Pattern playback).* Postoji više verzija ovog uređaja, ali samo je jedna poznata. Uređaj konvertira slike akustičnih uzoraka govora u formi spektrograma nazad u zvuk. Koristeći ovaj uređaj Alvin Liberman i njegove kolege bile su u stanju otkriti akustične tragove kod fonetskih segmenata.

Prvi računalni sistemi sinteze govora su stvoreni ranih 60 – tih godina. Godine 1961, fizičar John Larry Kelly Jr. i njegovi suradnici koristili su IBM 704 da sintetiziranju govor. Taj moment smatra se važnim događajem u Bell Laboratorijima.

Kellyjev glas je sintetizirao pjesmu "Daisy Bell". Usput rečeno, izvedba te pjesme od strane računala inspirirala je Arthur C. Clarka, koji je u to vrijeme posjetio Bell Laboratorij, da u svojoj noveli *Odiseja 2001 u Svemiru* podari glas imaginarnom računalu *HAL 9000*.

Dominantni sistem u 1980. i 1990. je bio **MITalk** sistem, baziran na radu Dennisa Klatta u MIT – u i **Bell Labs Speech System** koji je bio višejezični. Rani TTS sistemi su zvučali kao roboti i bili su često jedva razumljivi. Kvaliteta sintetiziranog zvuka se gradualno povećavala, ali ni danas nije na razni prirodnog ljudskog glasa.

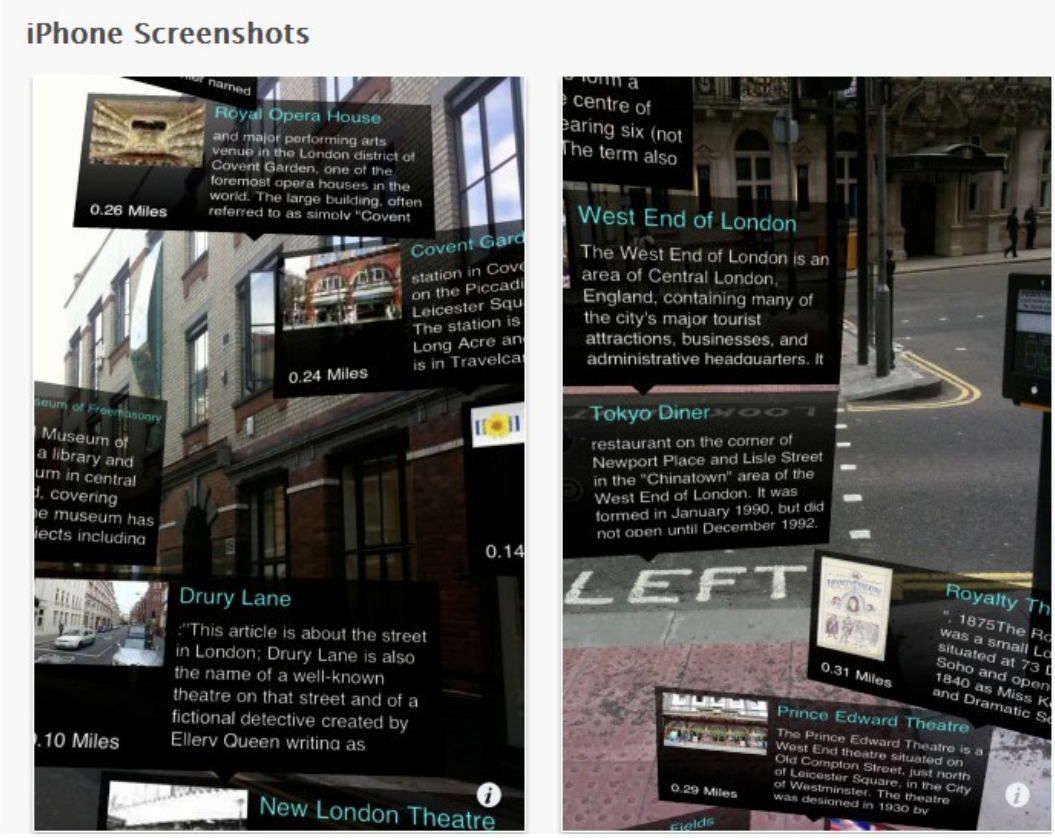
2 Opis rada korištenih tehnologija

2.1 Opis rada tehnologije AR – a kroz primjere

Video igre su usputna zabava još otkako je ranih sedamdesetih stvoren **PONG.** Od tada je računalna grafika postala kompleksnija do te mjere da najnovije igre prelaze zadnje granice foto-realizma. Kod takve tehnologije teško je pojmiti što je stvarno, a što računalno generirano.

2.1.1 Nearest Wiki aplikacija

Kod virtualne stvarnosti generiramo sav okoliš, nasuprot tome proširena stvarnost je bliža stvarnome svijetu. Proširena stvarnost dodaje grafiku, zvuk, miris i opis okolišu koji već postoji. Mobilne aplikacije koriste činjenicu da proširena stvarnost može opisati okoliš koji se snimi kamerom. Takve aplikacije su od velike koristi osobama kao turisti ili vojnici. Dobar primjer jedne takve aplikacije bila bi **Nearest Wiki** [13] mobilna aplikacija koja opisuje sadržaj mjesta koje snima.



Slika 4. Screenshot rada aplikacije *Nearest Wiki*

Podatci o poziciji kao fotografija mjesta, te GPS koordinate i smjer se šalju na server direktno povezan s **Wikipediom** koja vraća opis prepoznatih mjesta, te ih se prikazuje na ekranu mobitela.

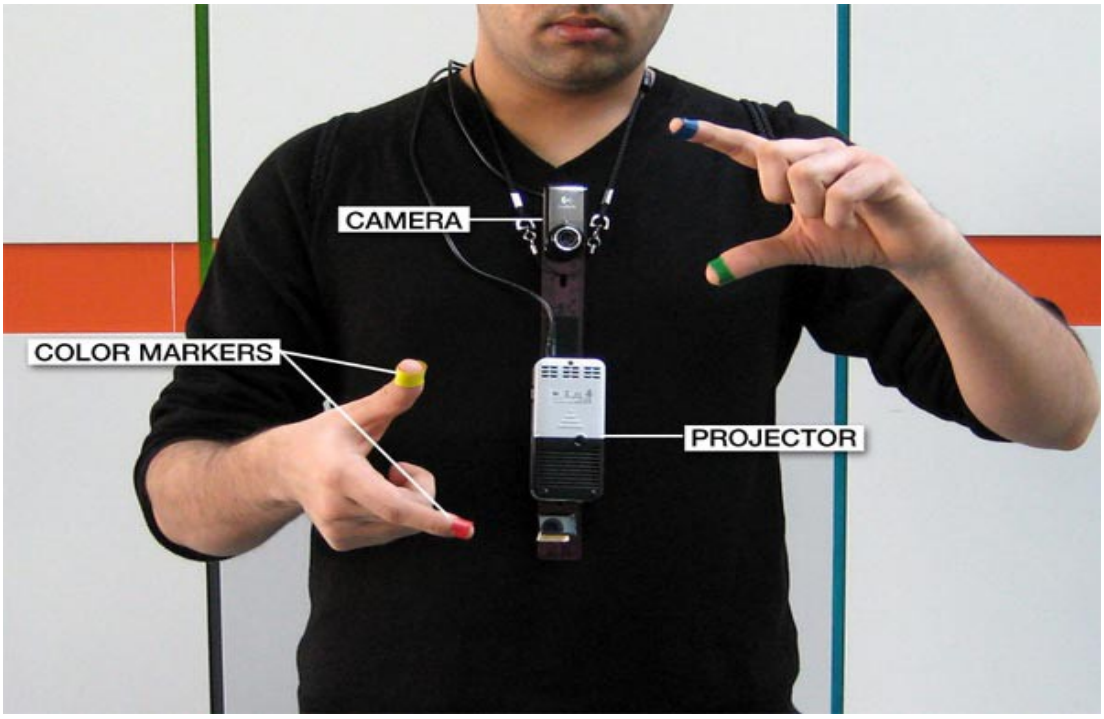
Osnovna ideja Proširene stvarnosti jest da se doda grafika, audio i ostala senzorna poboljšanja preko stvarnog okoliša. Iako zvuči prilično jednostavno (vremenska prognoza na televiziji radi isto već godinama) Proširena stvarnost je naprednija tehnologija nego ona koja se koristi pri prikazivanju efekta za vrijeme emitiranja.

2.1.2 SixthSense Aplikacija

Neke od najuzbudljivijih novih tehnologija baziranih na Proširenoj stvarnosti se razvijaju u Sveučilišnim istraživačkim laboratorijima. U veljači 2009, na TED konferenciji, Pattie Maes i Pranav Mistry iz *MIT Media Lab's Fluid Interfaces Grouo* predstavili su njihov AR sistem nazvan **SixthSense**[14] *(Šesto Čulo)* koji se oslanja na neke od osnovnih komponenti u AR sistemima.

- Kamera
- Projektor
- Pametni telefon
- Zrcalo

Ove komponente se koriste kao uređaj koji se nosi oko vrata. Korisnik također nosi na vršcima prstiju četiri različito obojane kape pomoću kojih manipulira fotografijama koje uređaj projektira.



Slika 5. Sixth sense uređaj

SixthSense koristi kameru, GPS uređaj i Internet pomoću kojih analizira okruženje. Uređaj pomoću tih jednostavnih komponenti projektira na bilo kakvu površinu ispred korisnika interaktivni ekran. Kako se kamera nosi na prsima, proširit će se ona stvarnost koja je ispred korisnika, tj. ona koju on vidi. Na primjer; ako je korisnik u trgovini, SixthSense može projektirati informacije o proizvodima kao: nutritivna vrijednost, cijena itd.

2.1.3 Layar Aplikacija

Iako će proći još vremena dok se aplikacija kao SixthSense ne pojavi na mobitelu, primitivne verzije proširene stvarnosti već postoje na iPhone – u i uređajima koji koriste Android OS. U Nizozemskoj vlasnici mobilnih uređaja su u stanju skinuti *(download)* aplikaciju zvanu **Layar**[15]**.** Aplikacija koristi kameru i GPS da skupi informacije o okruženju. Layar daje informacije o restoranima ili drugim objektima u blizini. Moguće je snimiti zgradu i Layar će provjeriti postoji li poslovni objekt unutar iste.

2.1.4 Yelp Aplikacija

Layar nije jedina aplikacija tog tipa. U Rujnu 2009 iPhone korisnici su se ugodno iznenadili kad su našli *iznenađenje* **"easter egg"** sakriveno unutar **Yelp**[16] aplikacije. Yelp korisnicama dobavlja ocjenu restorana i ostalih poslovnih objekata. Od 2009 sadrži AR komponentu zvanu **Monocle**, koji koristi GPS i kompas uređaje pomoću kojih također prikazuje informacije o lokalnim restoranima.

Osim gore navedenih aplikacija, još je mnogo drugih u fazi razvoja. S određenim poznavanjem korisničkih preferencija, vraćaju se informacije koje bi mu mogle biti zanimljive. Tehnologija nije još savršena, ali veliki broj ljudi radi na njoj.

2.1.5 ARToolKit

ARToolKit je softver, C/C++ knjižnica *(library)* koja se koristi za izgradnju aplikacija Proširene stvarnosti[17].

Jedan od najvećih izazova pri razvoju AR aplikacija je praćenje korisničkog položaja u odnosu na položaj gdje se mora iscrtati računalna grafika. Aplikacija mora znati poziciju korisnika da bi pravilno iscrtala imaginarne objekte.

ARToolKit korisiti algoritme Računalna vida *(Computer vision)* pomoću kojih rješavaju ovaj problem. ARToolKit u realnom vremenu računa položaj i orijentaciju kamere prema fizičkom okruženju koji omogućuje lakši razvoj različitih AR aplikacija. ARToolKit sadrži sljedeća svojstva:

- Praćenje pozicije i orijentacije kamere
- Praćenje i pronalaženje položaja geometrijskih oblika
- Lako kalibriranje svojstava kamere
- Dovoljno brze AR aplikacije
- Linux, MacOS i Windows distribucija
- Slobodan kod *(open source)*

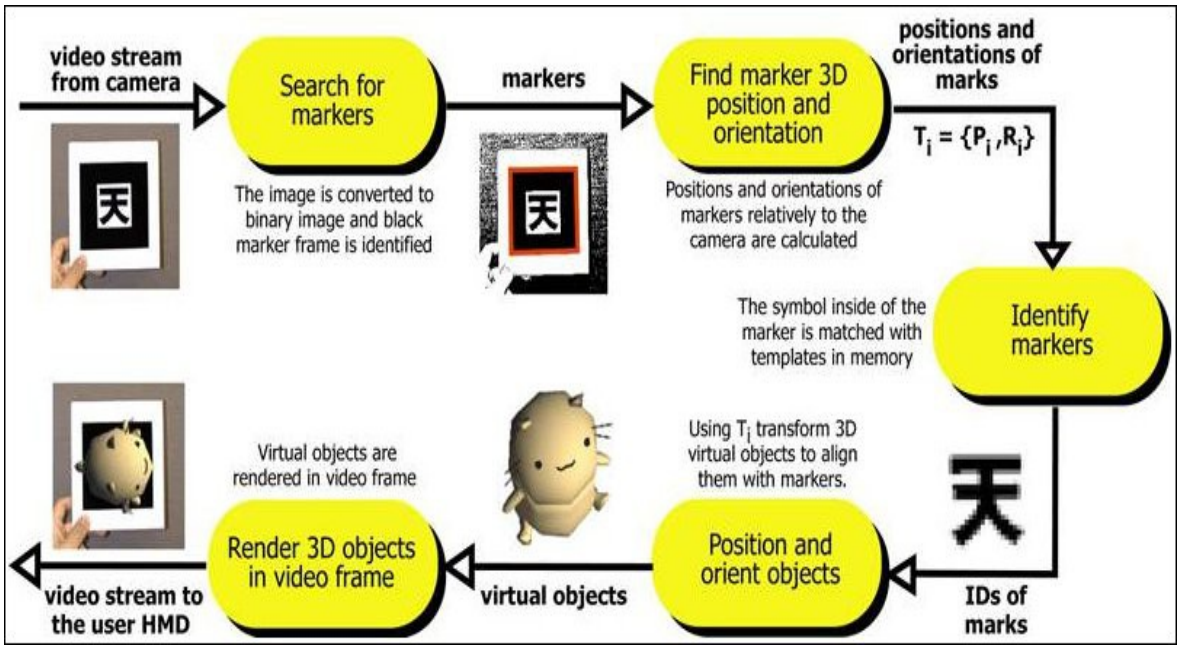
ARToolKit je originalni izum Dr. Hirokazu Kato. Njegov je razvoj podržan od

strane *Human InterFace Lab (HIT Lab NZ)* Sveučilišta Washington *(University of Washington)* i *HIT Lab NZ* Sveučilišta Canterbury, Novi Zeland *(University of Canterbury, New Zealand)* te ARToolworks Inc, Seattle[17].

ARToolKit aplikacija dodaje virtualne oblike unutar videa stvarnog svijeta. Proces rada se može opisati na sljedeći način:

1. Kamera snimi video stvarnog svijeta, te ga pošalje računalu
2. Računalni softver prolazi kroz svaku sliku videa *(frame)* i traži kvadratne oblike.
3. Ukoliko se pronađe navedeni oblik, ARToolKit izračunava poziciju kamere relativno prema njemu
4. Jednom kad se pozicija kamere izračuna, računalni grafički oblik se iscrtava na toj poziciji
5. Konačni izlazni podatci *(output)* prikazuju se na ekranu uređaja. Kad korisnik pogleda na ekran, vidi iscrtanu grafiku na mjestu kvadratnog oblika

Slika dolje demonstrira navedene korake. ARToolKit je u stanju pratiti poziciju kamere u stvarnom vremenu čime se osigurava konzistentnost pozicije objekta koji se prikazuje.



Slika 6. Dijagram rada ARToolKit knjižnice

Postoje određena ograničenja pri radu sa ARToolKit – om. Virtualni objekti se prikazuju samo onda kad su **oznake praćenja** *(tracking marks)* u potpunosti unutar vidokruga kamere. To znači, ukoliko cijeli uzorak nije vidljiv ARToolKit ga nije u stanju detektirati što često vodi do praznina pri pomicanju kamere. Naprednije (komercijalne) AR knjižnice kao **ARTag** (http://www.artag.net/) rješavaju i taj problem.

Postoje određeni problemi koji se tiču dometa. Što je veći uzorak koji se treba prepoznati, veća je maksimalna udaljenost s koje se može detektirati. Tabela 1. prikazuje tipične maksimalne domete detekcije s obzirom na veličinu.

Veličina uzorka (cm)	Domet detekcije (cm)
7	45
9	65
11	85
13	105

Tabela 1. Domet praćenja ovisno o veličini

Na domet također utječe kompleksnost uzoraka. Jednostavniji uzorak je bolji. Uzorci s velikim crno – bijelim regijama su najbolji.

Utječe još i orijentacija oznake praćenja relativno prema kameri. Kako oznake postaju više nakrivljene i horizontalne, manje je ukupnog uzorka vidljivo, pa je manja šansa da se uzorak prepozna.

Konačno, na rezultate praćenja utječe i okoliš, kao na primjer osvjetljenje. Svjetla na stropu mogu stvoriti refleksiju čime otežavaju detekciju. Da bi se refleksija izbjegla, uzorci se prikazuju na ne – refleksnom materijalu.

2.1.6 ARToolKit primjer

Nakon što se ARToolKit instalira na **Windows** OS – u ili se prevede *(compile)*

na Windows/Linux OS – u, za svrhe demonstracije postoji jednostavni program koji detektira jedan uzorak i zamjenjuje ga s računalno generiranom kockom. Program se nalazi unutar: *{instalacijska putanja}/ARToolKit/bin/simpleTest.o*

Uzorak se nalazu unutar:

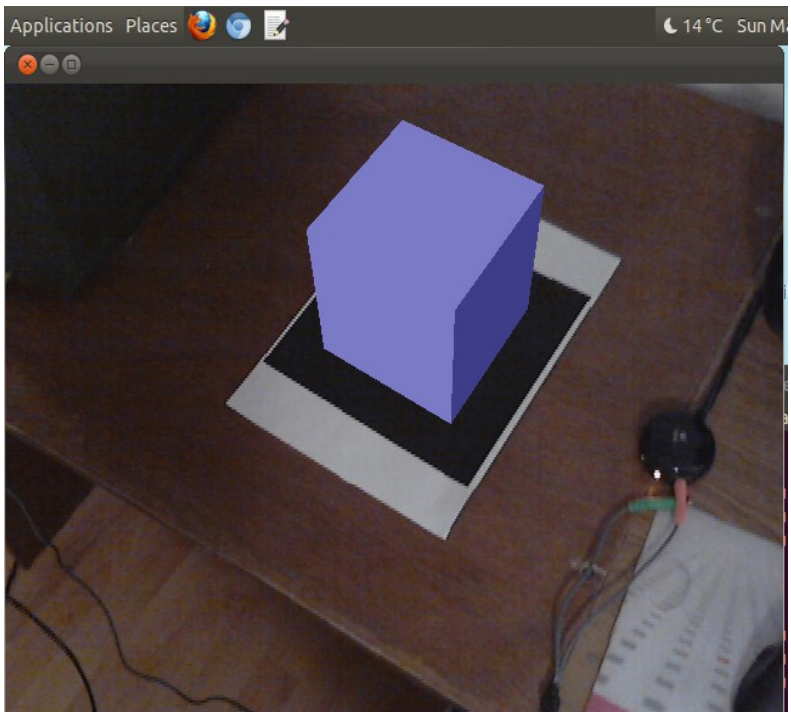
{instalacijska putanja}/ARToolKit/patterns/pattHiro.pdf

Slika 7. Uzorak pattHiro

Nakon pokretanja programa pokreću se upravljački programi *(drivers)* kamere, te se na ekranu otvara dijalog koji prikazuje sadržaj koji kamera snima.

Slika 8. Sadržaj kamere

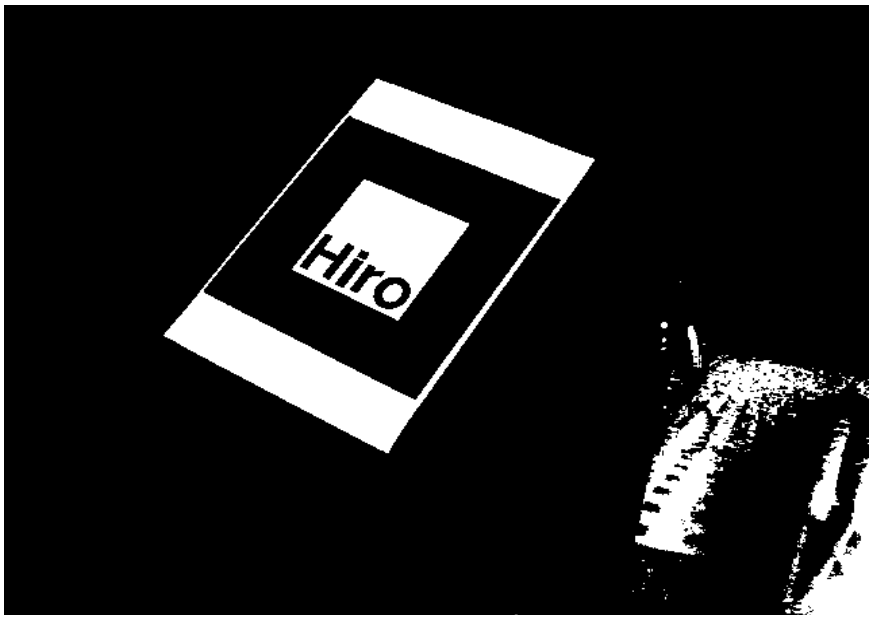
Na slici gore prikazan je navedeni sadržaj gdje se može vidjeti da se unutar video – kruga kamere nalazi uzorak pattHiro. Nakon par sekundi prikazani uzorak se zamjenjuje generiranom kockom.



Slika 9. Računalno proširena stvarnost

Da bi se virtualni objekt prikazao, cijeli crni okvir s uzorkom mora biti vidljiv. Ako se u kojem slučaju kamera pomakne toliko da uzorak nije u potpunosti vidljiv ili osvjetljenje nije dobro, objekt nestane. Napredniji AR okviri *(framework)* pamte pozicije uzoraka *(cache)* tako da je moguće prikazati virtualni objekt na parcijalnom uzorku ako se prije toga zapamtila njegova pozicija.

ARToolKit pri obradi slike videa koristi **Prag filtar** *(threshold filter)*. Filtar briše buku sa slike tako da je prikaže kao dvobitnu paletu. Piksel *(pixel)* na slici može biti ili crn ili bijel s obzirom na njegovu RGB vrijednost prema nekoj brojčanoj vrijednosti. Obično je ta vrijednost 100, ali može se mijenjati.



Slika 10. Slika nakon obrade Prag filtra

Usprkos navedenim nedostacima ARToolKit posjeduje i dosta prednosti; slobodan je i u potpunosti samoodrživ *(stand alone),* ponekad je dosta brz i precizan. Veliki broj istraživača koristi ga za eksperimentiranje koje je dokumentirano, što omogućuje drugim AR programerima kao i početnicima dosta dobru podršku.

3 OCR Tehnlogija

Tehnologija **Prepoznavanje optičkih znakova** *(Optical Character Recognition – OCR)* interpretira skenirane, tiskane znakove u računalni digitalni format. OCR koristi različite sisteme kao: identifikacija uzoraka *(pattern identification)*, umjetna inteligencija *(artificial intelligence)*, računalni vid *(machine vision*) da bi konvertirao skenirani tekst. Istraživanje ovog polja još napreduje, te daje sve bolje rezultate[20].

Dvije osnovne tehnike interpretacije karaktera koriste se pri transformaciji skeniranih dokumenata u računalni format:

1. **Usklađivanje matrica** *(matrix matching)*
2. **Izlučivanje uzoraka** *(pattern extraction)*

3.1 Opis i princip rada OCR – a

3.1.1 Usklađivanje matrica

Usklađivanje matrica (još poznato kao **usklađivanje uzoraka** – *pattern matching*) je jednostavnija i uobičajenija tehnika, ali je zato manje precizna. Radi na principu asociranja skeniranih oblika s pohranjenom kolekcijom predložaka *(template)*. Kad skenirani oblik odgovara jednom od ovih predložaka OCR ga identificira, te interpretira kao njemu odgovarajući ASCII ekvivalent. Nedostatak ove tehnike je da moraju postojati predlošci za sve moguće kombinacije veličina i stilova (fontova). Zbog toga je ovaj stil efektivan tek onda kad postoje unaprijed poznati stilovi i veličine slova[19].

3.1.2 Izlučivanje uzoraka

Izlučivanje uzoraka *(pattern extraction)* radi na principu identifikacije određenih svojstava individualnih znakova. Na primjer, malo slovo 't' je okarakterizirano s vertikalnom linijom koju pri vrhu siječe manja horiznotalna linija. Ovi uzorci ostaju manje – više jednaki, neovisno o stilu i veličini. Ipak, izlučivanje uzoraka može biti veoma složena kod određenih znakova. Dobar primjer bio bi engleski znak 'y' . Na slici 11. vidi se da postoje velike razlike kod oblika različitih

fontova, te je veoma teško za sve njih naći univerzalni uzorak[19].

Slika 11. Više različitih fontova istog slova

Većina OCR programa primjenjiva kombinaciju usklađivanja matrica i izlučivanja uzoraka koristeći usklađivanje matrica kod Courier fonta u kojeg su svi znakovi iste širine, a izlučivanje uzoraka za proporcionalne fontove kao Palatino i Helvetica.

3.1.3 Rukom pisani tekst

Određeni dokumenti se ne mogu konvertirati u elektronički format koristeći tradicionalni OCR jer su pisani rukom, a ne tiskani na stranicu. U tom slučaju potrebni su posebni OCR programi za detektiranje pisanog rukopisa. Kako je stil pisanja drukčiji od osobe do osobe, potrebna je sofisticirana analiza da bi se identificirali simboli. Prepoznavanje pisanog rukopisa također omogućuje alternativne metode unosa teksta u računalo. Ovakav način upisa, iako obećavajući, ujedno je i sporiji od tradicionalnog upisa pomoću tipkovnice. Iz tih razloga prepoznavanje pisanog teksta je ograničeno na uređaje kod kojih je zbog veličine, tipkovnica nepraktična[19].

Kao što različite veličine i stilovi fonta predstavljaju izazov pri detekciji tiskanog teksta, različiti stilovi pisanja također predstavljaju problem pri prepoznavanju rukom pisanog teksta. Da bi se povećala preciznost prepoznavanja individualnog rukopisa, većina OCR programa koristi neku tehniku **strojnog učenja** *(Machine Learning).* Pomoću ove metode softver uspoređuje primjere korisničkog rukopisa s tekstom koji se trenutno obrađuje umjesto da koristi generalne fiksne uzorke. Ako je korisnik konzistentan pri korištenju istog stila, ova metoda može biti veoma efektivna pri prepoznavanju individualnog rukopisa.

3.1.4 Alati korišteni pri procesuiranju slika

Da bi se izlučili simboli iz milijuna piksela unutar slike dokumenta, svaka komponenta OCR sistema je dizajnirana da smanji količinu nepotrebnih podataka. Prvi i najvažniji korak kod procesuiranja slike je određivanje područja interesa i čišćenje buke kako bi se što efikasnije izlučili potrebni podatci. Ovo poglavlje opisuje najuobičajenije tehnike kod takvih procesa, tj. jezgru *(kernel)* OCR sistema. Kreće s inteligentnim procesuiranjem **formi** (slika), pomoću kojih se određuje područje interesa. Nakon toga opisuje kako odvojiti znakove od pozadine, te kako ispraviti defekte nastale pri obradi[20].

3.2 Generički sistem obrade formi

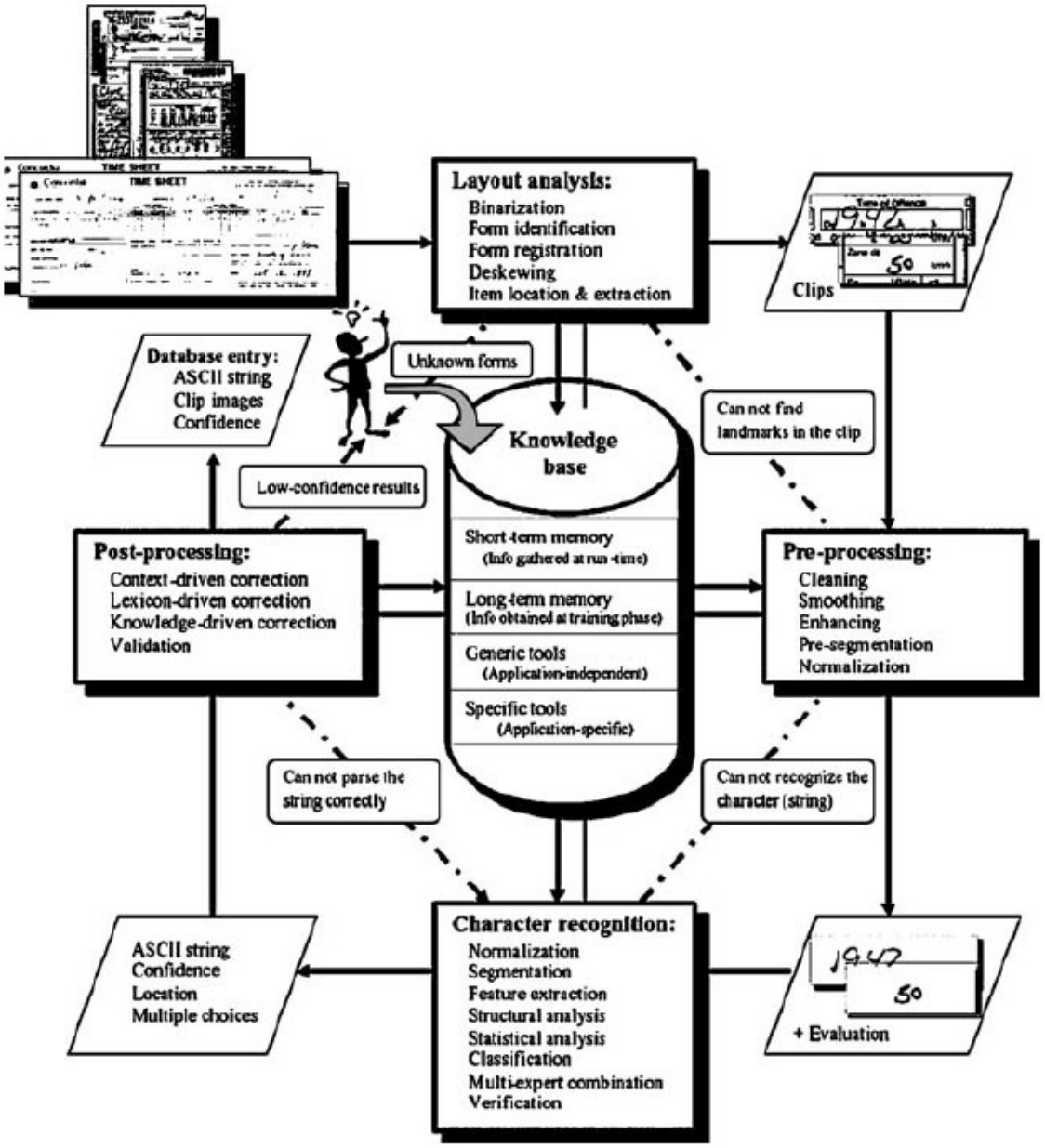
Ulazni podatci kod tipičnih procesnih sistema su obično slike u boji ili različite nijanse sive, kod kojih jednoj linearnoj liniji odgovara približno stotinu piksela. Izlaz odgovara ASCII znakovnom nizu *(String)* koji na memoriji računala zauzima par stotina **bajtova** *(byte)*. Da bi se postigao visok stupanj apstrakcije podataka, procesni sistem ovisi o sljedećim koracima:

1. **Binarnizacija** *(Binarization)*: pretvaranje slike u binarni format
2. **Dobivanje slika** *(Image acquisition)*: dobivanje slike forme u boji, u nijansama sive ili u binarnom formatu
3. **Indentifikacija formi** *(Form identification)*: za danu sliku forme odrediti najefikasniji model
4. **Analiza konstrukcije** *(Layout analysis)*: razumijevanje strukture forme i semantičkog značenja informacija unutar iste
5. **Izlučivanje podataka** *(Data Extraction):* izlučivanje podataka iz odgovarajućih polja i procesuiranje podataka da bi se isti bolje označili
6. **Prepoznavanje znakova** *(Character recognition)*: pretvorba podataka koji sadrže tekstualne znakove u njima odgovarajući digitalni format što uključuje i analizu sintakse

Svaki od gore navedenih koraka smanjuje količinu informacija potrebnih da bi

se obradio sljedeći korak. Konvencionalni pristup se sastoji od prolaza informacija kroz ove komponente, te odabira najboljeg rješenja za svaki korak čime smo stvorili inteligentan sistem procesuiranja formi. U srcu tog sistema nalazi se **Baza znanja** *(Knowledge database)* – skup informacija skupljenih pri radu OCR sustava koje pomažu pri odabiru metoda i alata.

Pregled inteligentnog sistema obrade formi s tipičnim ulazima i izlazima ilustriran je na slici 12. Jezgra ovog inteligentnog sistema obrade formi sastoji se od **kratkotrajne** *(short – term)* i **dugotrajne** *(logn – term)* memorije, te seta generičkih i specijaliziranih alata za procesuiranje dokumenata. Kratkotrajna memorija pohranjuje znanje stvoreno za vrijeme izvođenja programa *(run – time).* Dugotrajna memorija pohranjuje znanje stvoreno za vrijeme **trening faze** *(traning phase)*, kao na primjer reference za prepoznavanje karaktera i logotipa *(log),* te identifikacija tipova formi. Drukčiji tipovi formi i slika mogu se okarakterizirati po njihovom specifičnom formatu *(layout).* Umjesto čuvanja originalnih slika za modele formi, trebamo sačuvati samo izlučne **potpise** *(signatures)* da bi izgradili bazu poznatih tipova formi.



Slika 12. OCR sistema obrade formi

Potpis slike forme je definiran kao skup statističkih i strukturalnih svojstva izlučnih iz ulaznih podataka (ulaznih slika). Generički alati uključuju različite binarnizacijske metode kao na primjer metode normalizacije. Specijalizirani alati sadrže metode prepoznavanja, čišćenja, verifikacije itd. Kad ulazni dokument prođe kroz OCR sistem informacija, baza znanja pomaže pri odabiru gore navedenih metoda. Baza znanja za vrijeme izvođenja rada OCR sistema skuplja informacije o analizama ulaznih dokumenata, te je na temelju tih podataka u stanju prepoznati slične forme.

3.2.1 Model Obrade za eliminaciju kompleksne pozadine

Prije nego što se podatci od interesa prepoznaju i konvertiraju u jednostavne ASCII nizove, važan korak je izlučiti samo piksele što pripadaju znakovima koji se prepoznaju. Kod dokumenata koji sadrže dosta kontrastnih boja što se razlikuju od crne tinte korištene pri ispisu alfa – numeričkih znakova, možemo jednostavno koristiti filtre boja. Nažalost, veliki broj dokumenata zbog niže cijene ispisa koristi monokromatski format ili ispisuje u nijansama sive boje. Osim toga proces obrade slika je vremenski skup.

Kod dokumenata koji sadrže nijanse sive boje, često se koristi binarnizacija slika pomoću čega se minimizira šum i konvertira slika u monokromatski format. Preciznost izlučnih podataka je kritična kod prepoznavanja znakova. Varijabilnost pozadine i struktura dokumenata, zajedno s kompleksnosti oblika, čini algoritme i strategije automatizacije veoma kompleksnim.

Postoje različite tehnike izlučivanja znakova. Sljedeće činjenice se uzimaju u obzir pri odlučivanju koju tehniku odabrati:

- Znakovi se sastoje od **Linija**
- Linija je izdužena i povezana komponenta koja je ili tamnija ili svjetlija nego njezino okruženje
- Linija je nominalno konstantne širine definirane kao dvije tranzicije sive boje u neku drugu
- Boja pozadine linije sadrži dosta manju varijaciju od linije

Najefikasniji način izlučivanja piksela znakova je kroz intenzitet boja, tj filtar praga. Filtar praga odvaja piksele pozadine od znakova tako da uspoređuje RGB vrijednosti nijansi sive prema nekoj varijabilnoj vrijednosti. Binarnizacijske tehnike bazirane na modelu praga mogu se podijeliti u dvije kategorije:

- **globalni algoritam praga** *(global threshold)*
- **lokalni algoritam praga** *(local threshold)*.

3.2.2 Globalni algoritam praga

Globalni algoritam praga koristi jedinstveni prag za cijelu sliku. Piksel koji ima RGB vrijednost višu od zadane postaje crn (255, 255, 255 RGB vrijednost), a koji ima vrijednost manju od zadane postaje bijel (0, 0, 0 vrijednost).

Kod dokumenata čiji znakovi sadrže konstantnu razinu sive boje, koja je također prisutna kod svih pozadinskih objekata, globalne metode praga su najefikasnije. Histograf ovih slika je bi-modalan tako da znakovi i pozadina jednako doprinose distinktivnim vrijednostima, te se globalni prag može odabrati na način kojim se selektiraju samo pikseli znakova.

Među mnogim tehnikama pronalaženja praga najefikasnija je **Otsu metoda**. Otsu metoda (još znana kao **optimalni prag** - *optimal thresholding*) dijeli se na dvije klase; **C0** i **C1** te pragu **t**. Prag t se pronalazi tako da se minimizira greška klasifikacije piksela bez da se izgubi svojstvo generalizacije. Objekte definiramo kao tamne znakove naspram svijetle pozadine. Kod slike sa razinama sive boje definiranim kao $G=\{0,1,\ldots,L-1\}$, objekt i pozadina mogu biti definirani kao $C0=\{0,1,\ldots,t\}$ i $C1=\{t+1,t+2,\ldots,L-1\}$. Elegantan i jednostavan kriterij pri pronalaženju optimalnog praga klasa je **raspršenost** *(variance).* Raspršenost je statistički pojam koji predstavlja odstupanje od medijana, te se može podijeliti na raspršenost unutar klasa, između klasa, te totalna.

- σ_W^2 - Raspršenost unutar klasa (1)
- σ_B^2 - Raspršenost između klasa (2)
- σ_T^2 - Raspršenost totalna (3)

Prag t se nalazi maksimiziranjem jedne od tri navedene funkcije.

$$\lambda=\frac{\sigma_W^2}{\sigma_B^2} \quad , \quad \eta=\frac{\sigma_B^2}{\sigma_T^2} \quad , \quad \kappa=\frac{\sigma_T^2}{\sigma_W^2} \tag{4}$$

Kod koje:

$$P_i = n_i / n \quad , \quad w_0 = \sum_{i=0}^{t} Pi \quad , \quad w_1 = w - w_0 \tag{5}$$

$$\mu_T = \sum_{i=0}^{L-1} iPi \quad , \quad \mu_t = \sum_{i=0}^{t} iPi \quad , \quad \mu_0 = \frac{\mu_t}{w_0} \quad , \quad \mu_l = \frac{\mu_T - \mu_t}{l - \mu_0} \tag{6}$$

$$\sigma_T^2 = \sum_{i=0}^{L-1} (i - \mu_T)^2 Pi \quad , \quad \sigma_B^2 = w_0 w_1 (\mu_1 - \mu_0)^2 \tag{7}$$

n_i je *i* – ti element histografa, tj broj piksela na sivoj razini *i.* (8)

$n = \sum_{i=0}^{L-1} n_i$ je ukupni broj piksela na slici. (9)

$P_i = n_i / n$ je vjerojatnost ponavljanja na sivoj razini *i* (10)

w_0 i w_1 predstavljaju dijelove slike koje zauzimaju pozadina i objekt klase (11)

Maksimalna vrijednost η , još predstavljena kao η^* može poslužiti kao mjerilo različitosti između klasa, tj. bimodalnost histografa. Takvu vrijednost uzimamo kao prag.

3.2.3 Lokalni algoritam praga

Lokalni algoritam praga *(Local Gray Level Thresholding)* – dokumenti koji su dizajnirani sa kompleksnim i obojanim pozadinama predstavljaju problem izlučnim metodama. Dok globalni algoritam praga radi zadovoljavajuće kod monokromatskih formi, lokalni algoritmi praga su u stanju dati bolje podatke kod kompleksnijih pozadina. Nedostatak ovog pristupa je duže vrijeme obrade.

Komparativna studija binarnizacijkih metoda je od 11 najpoznatijih algoritama izdvojila tri najefikasnija: *Yanowitz i Bruckstein metoda, Niblack,* te *Bernsen metoda.*

Od navedenih 11 metoda, 8 je koristilo filtar praga, a ostale 3 je izlučivalo

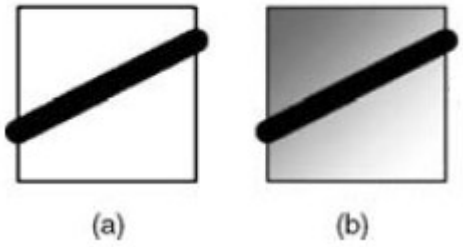
piksele znakova nakon što je detektiralo njihove rubove[22][23][24].

Dodatni proces pri radu navedenih metoda je uklanjanje objekta **duhova** *(ghost)*. Prosječna vrijednost gradijenta ruba je izračunata, te objekti koji imaju prosječan gradijent ispod praga su označeni kao pozadina, te maknuti.

Bernsen metoda izračunava lokalni minimum i maksimum gradijenta "susjedstva" oko svakog piksela $f(x,y) \in [0, L-1]$, te koristi njihov medijan kao prag.

$$b(x,y) = if\ f(x,y) < g(x,y)\ then\ 1\ otherwise\ 0 \qquad (12)$$

$F_{max}(x,y)\, i\, F_{min}(x,y)$ su maksimalne i minimalne vrijednosti gradijenta oko svakog piksela.



Slika 13. Linije na dvije vrsta pozadina

Slika 13. prikazuje linije na dvije vrste pozadina, (a) jednostavna pozadina, koju je lako binarnizirati pomoću algoritma globalnoga praga, (b) kompleksnija pozadina, koja se binarnizira pomoću algoritma lokalnog praga.

Da bi se izbjegao fenomen "duha" **lokalna raspršenost** *(local variance)* $c(x,y)$ se može izračunati kao razlika između maksimalne i minimalne vrijednosti gradijenta oko svakog piksela:

$$c(x,y) = F_{max}(x,y) - F_{min}(x,y) \qquad (13)$$

Klasifikacije piksela kao objekt ili pozadina se verificira prema sljedećim uvjetima:

$$b(x, y) = \begin{cases} 1 & \text{if } (f(x, y) < g(x, y) \text{ and } c(x, y) > c^*) \text{ or} \\ & (f(x, y) < f^* \text{ and } c(x, y) \leq c^*), \\ 0 & \text{otherwise.} \end{cases}$$

Prag c^* i f^* se nalaze kroz Otsu metodu koristeći histograf $c(x,y)$ i $g(x,y)$.

Ova metoda daje zadovoljavajuće rezultate kod slika sa nevarirajućim pozadinama, tj s pozadinama koje ne prelaze naglo iz jedne boje u drugu.

Slika 14. Varirajuća pozadina

Slika 14. prikazuje objekt s oštrim rubovima i naglim prijelazima, očito neodgovarajuću kod primjene navedenog algoritma. Kod takvih slika potrebni su kompleksniji i precizniji algoritmi da bi se dobili primjenjivi rezultati.

3.2.4 Odabir najefektivnije metode izlučivanja znakova

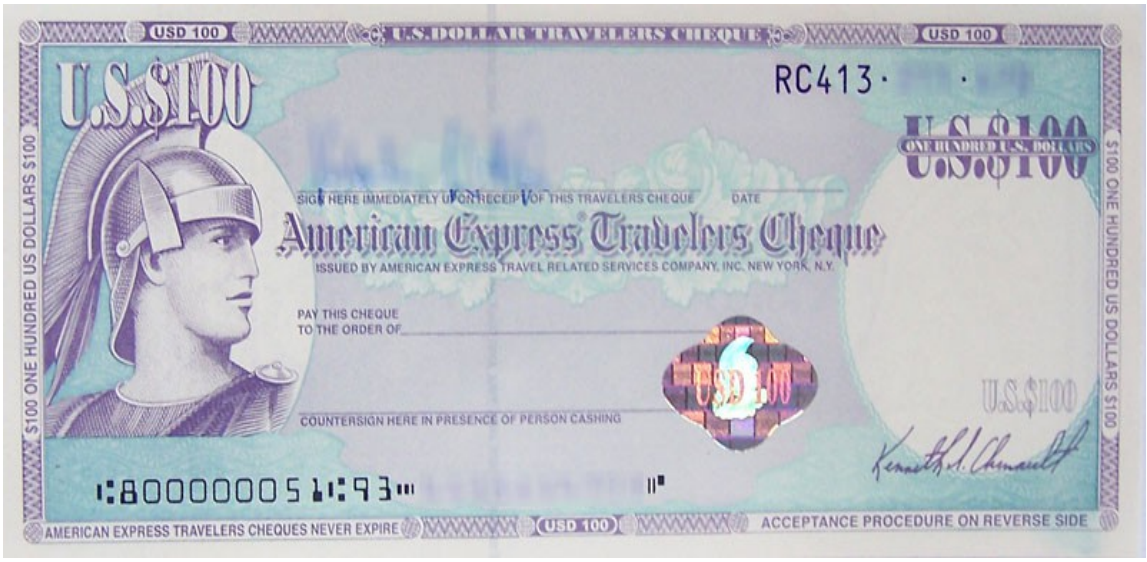
Efikasnost izlučivanja znakova ovisi o kompleksnosti pozadine forme. Dokumenti često sadrže stilski bogate i kompleksne pozadine što otežava rad OCR sistema. Poseban problem nastaje pri odabiru najbolje metode (algoritma) izlučivanja znakova.

Iako tehnike globalnog praga daju sasvim zadovoljavajuće rezultate kod jednostavnih "službenih" dokumenata, te stranica knjiga, tehnike lokalnog praga eliminiraju nepotreban šum kod kompleksnijih formi. Dokumenti, kao ček na slici 15,

sadrže dosta nepotrebnih podataka kao na primjer "linija" iznad kojeg se upisuje ime uplatitelja. Kod takvih formi najbolje je koristiti lokalne algoritme praga, čime smo eliminirali nepotrebne podatke.

Slika 15. Jednostavni Bankovni Ček

Nasuprot tome lokalni algoritam praga ima dulje vrijeme obrade, te nije optimalan kod dokumenata čije pozadine sadrže oštre rubove i nagle prijelaze (kontraste) boja, kao ček prikazan na slici 16. Kod takvih formi potrebni su kompleksniji algoritmi te sukladno tome i duže vrijeme obrade.



Slika 16. Kompleksni Bankovni Ček

Odabir algoritma, tj. metode obrade, bazira se na pronalaženju broja operacija za izračunavanje praga i analize slike. Prolazeći kroz sve piksele od kojih se slika

sastoji, dobivaju se statističke informacije o kompleksnosti pozadine. Na temelju dobivenih informacija OCR aproksimira broj potrebnih operacija da bi se dobio prag, te vrijeme izvođenja obrade. Na kraju se odabire ona metoda koja daje najbolji odnos broja operacija/vrijeme izvođenja.

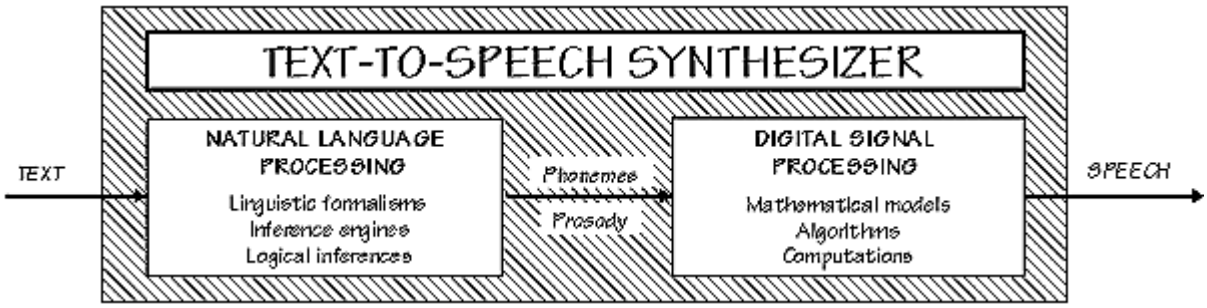
4 Opis i princip rada tehnologije TTS – a

TTS je računalni sistem koji je u stanju na glas pročitati tekst. Slika 17. prikazuje uobičajeni TTS koji se sastoji od[26][28]:

1. **Prirodna obrada signala** *(Natural Language Processing – NLP)*
2. **Digitalna obrada signala** *(Digital Signal Processing – DSP).*

NLP je u stanju proizvesti transkript fonema teksta koji se obrađuje zajedno sa ritmom i intonacijom.

DSP transformira simbolične informacije koje primi u govor. Izvođenje navedenih koncepta varira u primjeni algoritama i formalizama ovisno o jeziku koji se sintetizira.



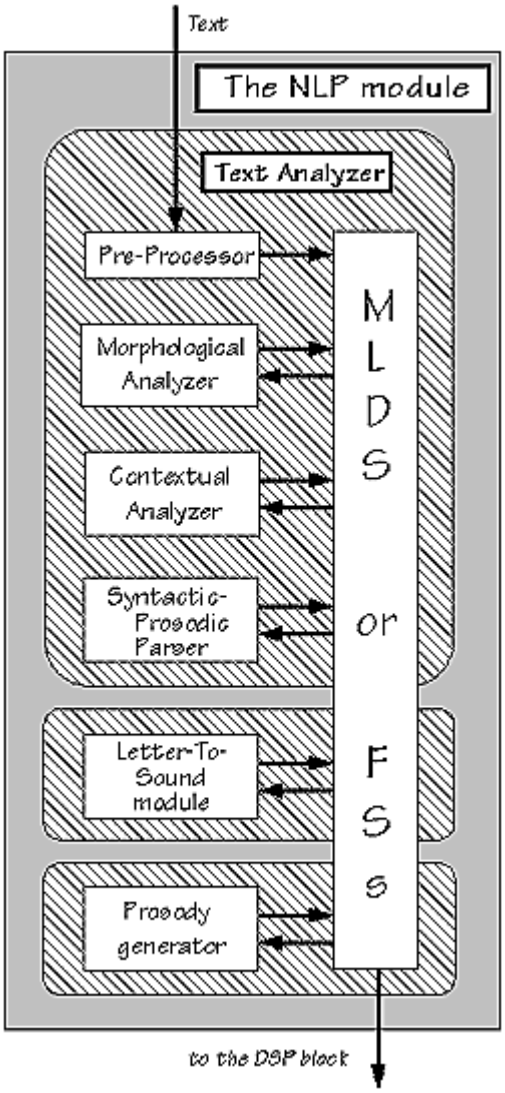
Slika 17. Uobičajeni TTS

4.1 NLP modul

NLP modul se sastoji od[26][28]:

- Pre – Procesor *(Pre – Processor)* Modula
- Morfološki Analizatora *(Morphological Analyzer)*

- Analizatora Konteksta *(Contextual Analyzer)*
- Sintaksno – Prozodijski Analizator *(Syntatic – Prosodic Parser)*
- Slovo u Zvuk modul *(Letter to Sound Module)*
- Prozodijski generator *(Prosody Generator)*



Slika 18. Kostur NLP modula

4.1.1 Pre – Procesor

Pre – Procesor modul organizira ulazne rečenice u liste riječi. On identificira brojke, kratice, akronime i idiomatiku, te ih transformira po potrebi. Poseban problem predstavlja nejasnost razine naglaska, pogotovo na kraju rečenice. Djelomično ga se rješava gramatičkom alanlizom.

4.1.2 Morfološki Analizator

Svrha Morfološkog analizatora jest analizirati svaku riječ zasebno. Riječi se rastave na elementarne jedinice (morfem), te se analizira njihova intonacija.

4.1.3 Analizator Konteksta

Analizator konteksta analizira kontekst skupa riječi, što smanjuje listu mogućih kategorija. To se može omogućiti pomoću n-grafa koji opisuje lokalne sintaksne ovisnosti u obliku determinističkog automata ili pomoću strojnog učenja tj. treniranjem klasifikacijskih stabala.

4.1.4 Sintaksno – Prozodijski Analizator

Sintaksno – Prozodijski Analizator provjerava razmake *(blank space)* i pronalazi strukturu teksta pomoću čega se bolje određuje intonacija rečenice.

3.1.5 Slovo u Zvuk modul

Slovo u zvuk modul *(LTS – letter to sound module)* je odgovoran za automatsku fonetsku transkripciju. Da bi se ostvarila transkripcija koriste se **fonetski rječnici** *(fonetic dictionaris)* i **rječnici izgovora** *(Pronunciation dictionaries)*. Organizacija LTS modula se može ostvariti na više načina, ali najuobičajenije strategije su:

- strategije bazirane na rječnicima *(dictionary-based)*
- strategije bazirane na pravilima *(rule-based)*

Strategije bazirane na rječnicima se temelje na pohranjivanju maksimalnog fonološkog znanja u leksikone. Da bi se veličina leksikona održala relativno malom,

unosi su ograničeni na morfeme. Ukoliko se traženi morfem ne može pronaći u leksikonu, onda se za transkripciju koriste pravila. Ovaj pristup se koristi kod MITTALK sistema čiji se leksikonski rječnik sastoji od 12000 morfema. AT&T Bell Laboratorij TTS sistem također koristi leksikonski rječnik sa čak 43000 morfema.

Strategije bazirane na pravilima se temelje na setu pravila pomoću kojih se preslikavaju slova u zvuk (grafem u fonem). Iznimke predstavljaju riječi koje se izgovaraju na specifičan način, za koje postoje individualna pravila. Većina iznimki se odnosi na riječi koje se najčešće upotrebljavaju tako da realno malen rječnik iznimki može pokriti veliki postotak riječi.

Nekad se smatralo da su zbog bogatih fonetskih rječnika, strategije bazirane na rječnicima u stanju postići veću preciznost od Slovo u zvuk pravila. Nasuprot tome, veliki su se napori uložili u dizajniranje kompleksnih skupova pravila i iznimki. Najefikasnija primjena strategije ovisi o jeziku koji se sintetizira.

4.1.6 Prozodijski generator

Prozodija podrazumijeva melodiju i brzinu izgovora kao i naglašavanje riječi. Najočigledniji efekt prozodije je **fokus**[26][28]**.** Kad želimo naglasiti određeni slog, postoje određene fluktuacije u visini intonacije . Time je moguće naglasiti važnost određenih riječi, te izmijeniti kontekst rečenice (ironija, sarkazam, urgentnost itd..).

Prozodijski generator segmentira rečenicu u grupu slogova, nakon čega se identificiraju veze između njih. Ovo grupiranje je često organizirano u hijerarhiju pomoću koje se konstruira moguća prozodija.

Kvaliteta navedene prozodije je daleko od idealne. Zbog toga generirani zvuk najčešće primjenjiva neutralnu boju glasa, a prozodijska obrada služi da ne zvuči previše hladno i robotski.

4.2 DSP komponenta

DSP pomoću fonetskih jezičnih specifikacija prevodi u govor podatke koje je generirao NLP. Glavni izazov DSP – a je proizvesti govor koji je razumljiv i prirodan. Da bi se to ostvarilo koriste se dvije metode[25]:

Sinteza formata *(Formant Synthesis)* koristi akustični model da bi na temelju računalno generiranih zvukova proizveo ljudski govor. Ova metoda tipično proizvodi razumljiv, ali ne baš "prirodan" govor. Najpoznatiji primjer bio bi **MS Sam**. Iako nije primjenjiva kod većine sistema, ima prednost da se lako implementira, pa se većinom koristi kod igračaka ili tehnologije gdje estetika nije prioritet.

Konkantna Sinteza *(Concatenative Synthesis)* – snimka ljudskog glasa se razlomi u akustične jednice: foneme, slogove, riječi, fraze i rečenice, koje se pohrane u **bazu podataka** *(database).* Procesor dohvaća akustične jedinice iz baze i spaja ih (konkatenacija) na način kojim se dobiva najkvalitetniji govor.

4.2.1 Sinteza formata

Sinteza formata još znana kao **Sinteza na temelju pravila** *(Rule-based synthesis)* je kognitivni i generalni fonetski mehanizam. Sinteza formata prikazuje govor kao 60 parametara koji se odnose na format i frekvenciju prosječnog ljudskog glasa. Sam glas se generira na temelju pravila dobivenih analizirajući ljudski govor. Do sada razvijeni sistemi daju nezadovoljavajuće rezultate što se tiče prirodnosti govora. Bolja kvaliteta glasa i veća prirodnost je teoretski moguća, ali je potrebna daljnja analiza i veći broj pravila[26][28].

Sinteza formata ima potencijal da s vremenom postane efektivna sinteza govora, pa se zbog toga integrirala u velik broj TTS sistema.

4.2.2 Konkantna Sinteza

Spajanjem više manjih zvučnih jedinica u glas nastaju potencijalne greške. Greška se može dogoditi jer ne postoji snimljena verzija intonacije svake moguće riječi. Osnovna strategija je odabrati najuobičajenije fraze ili rečenice čime se barem smanjuje mogućnost greške[26][28].

Generirani govor najboljih konkantnih sistema je često teško razlikovati od pravog ljudskog glasa. Maksimalna prirodnost zahtjeva veliku bazu podataka snimljenog glasa, što je veća baza bolja je kvaliteta. Tipična TTS baza sadrži 100 do 200 MB.

4.2.3 Pripremanje baze fonema

Prije nego se proizvede govor, mora proći serija preliminarnih stadija . Prvo se odabiru segmenti na način da se minimiziraju budući konkatenacijski problemi. Fonemi se odaberu kao osnovne jedinice govora. Nakon što se sastavi potpuna lista segmenata, stvara se lista riječi na način da se svaki segment pojavi barem jednom.

Kako su segmenti koji se povezuju u lanac snimljeni kao jedinstveni fonetski izrazi, često nastaju nejednakosti u amplitudi izraza. Da bi se riješio taj problem, sintetizirani govor sadrži konstantnu intonaciju, tj. neutralnu boju glasa. Razine intonacije segmenata se modificiraju na način kojim se eliminiraju radikalne razlike u amplitudi. Razina navedene intonacije se postavlja na prosječnu vrijednost[26][28].

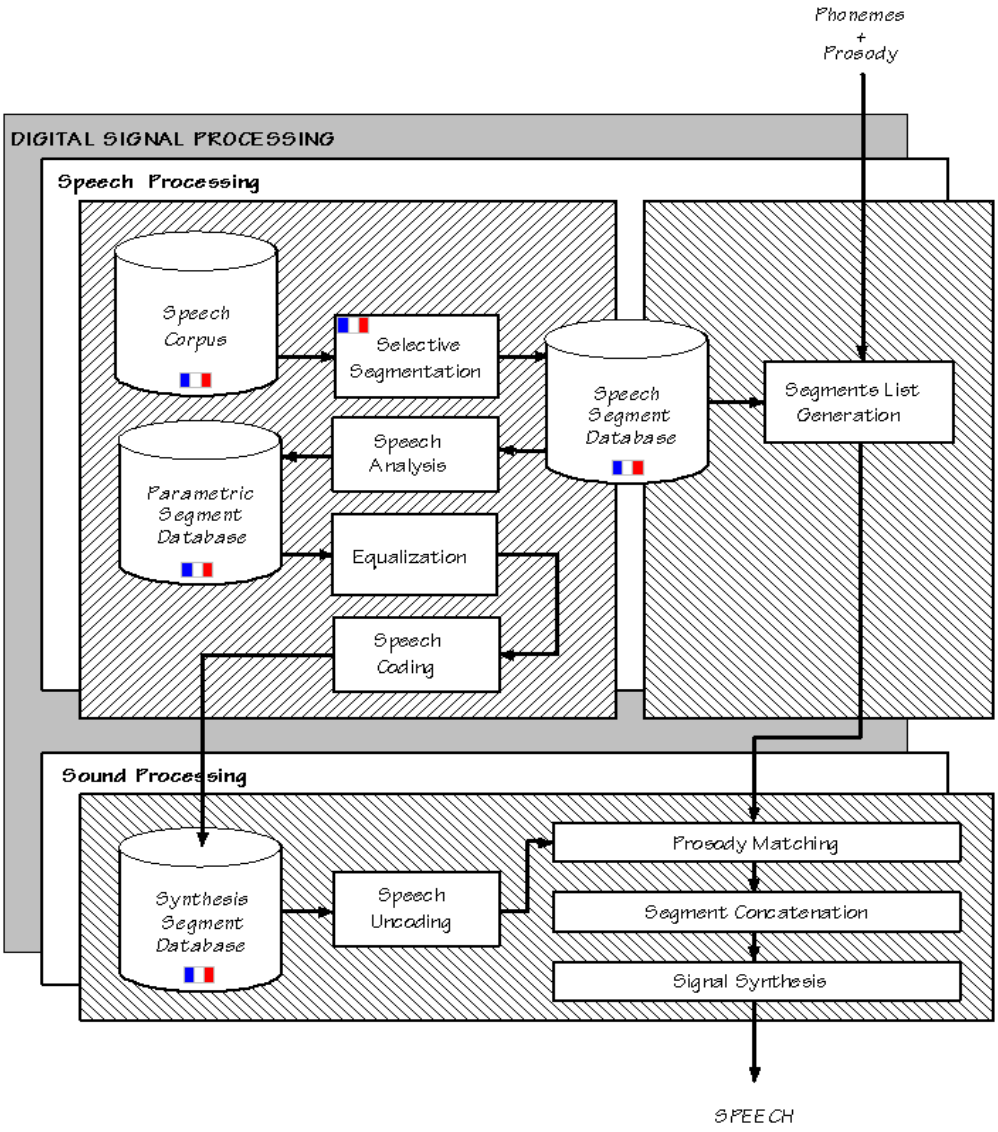
4.2.4 Pripremanje baze fonema za hrvatski jezik

Za potrebe sustava izrađena je baza fonema za hrvatski jezik. Prije izrade baze definiran je skup fonema koje će sustav razlikovati. Osim 30 standardnih fonema za hrvatski jezik dodatno se razlikuju naglašene inačice samoglasnika (a:, e:, i:, o: i u:), pojava glasa r u funkciji samoglasnika, tišina kao poseban fonem ili sveukupno 37 fonema. Na temelju toga skupa teorijski je moguće razlikovati 37*37=1369 fonema koji odgovaraju svim kombinacijama parova zadanih fonema[27].

Za izradu baze upotrijebljeni su unaprijed prikupljeni govorni zapisi jednog govornika iz hrvatskog govornog korpusa koji je sastavljen pri Odsjeku za informatiku Filozofskoga fakulteta u Rijeci. Korišteni su isječci snimljenih radijskih emisija informativnoga sadržaja – vijesti i vremenske prognoze. Format zapisa je jedno – kanalni .wav, frekvencije uzorkovanja 16 kHz. Ukupni broj zapisa je 2331, veličina 231 MB, a trajanje 2 sata i 26 min. Zvučne datoteke prethodno su bile obrađene postupkom automatskog prepoznavanja govora, čime su generirane .lab datoteke s oznakama izgovorenih fonema i automatski određenim trenucima početka i kraja svakog fonema izraženim u milisekundama. Uz zvučne datoteke i oznake dostupni su bili i prijepisi govora, te fonetski rječnik s 9922 unosa.

Dobiveno je ukupno više od 32 000 zapisa. Baza je na kraju svedena na minimalni oblik, tako da je svaki fonem predstavljen samo jednom inačicom. Ako je moguće, bira se fonem izrezan iz sredine riječi i prosječnog trajanja. Ako takav ne

postoji, uzima se u obzir samo trajanje. Rezultirajuća baza ima 923 unosa u .wav formatu uz pripadajuću tablicu pomoću koje se povezuju oznaka fonema i ime pripadne datoteke, vremenske oznake za svaki fonem i trenuci prijelaza između glasova u fonem. Veličina baze na disku je 2.47 MB.3.2.5 Sinteza govora *(Speech synthesis)*



Slika 19. Sinteza govora

Sekvenca segmenata se prvo dobije iz bloka **Generator liste segmenata** *(Segment List Generator)* prikazanog na slici 19. koji služi kao sučelje između NLP i DSP modula. Kad je prozodija određena i dodijeljena induvidualnim segmentima, stadij zvan **Komparator Prozodije** *(Prosody Matching)* šalje upite **Bazi Podataka**

Sinteze Segmenata *(Synthesisi Segment Database)* da bi se dobili zvukovi koji će se koristiti u sintezi glasa. Dobiveni zvukovi se prilagode odgovarajućoj im prozodiji. Blok **Konkatenacija Segmenata** *(Segment Concatenation)* spaja segmente i "izglađuje" nepravilnosti nastale pri konkatenaciji. Konačni rezultat se napokon proslijedi bloku **Sinteze Signala** *(Signal Synthesis),* čiji je zadatak da proizvede govor[26][28].

5 Praktični dio

Do sada je opisan princip rada svake individualne tehnologije. Sljedeći korak je primjena navedenih tehnologija unutar mobilne platforme.

Razvojem mobilnih uređaja otvaraju se nove mogućnosti primjene. Novi **pametni telefoni** *(smartphone)* posjeduju najsnažnije do sad procesore, visoko razlučive kamere, te vezu prema internetu. Velika prednost je da su osim toga još i mobilni. Do sad je bilo kakva primjena OCR – a, TTS – a ili AR – a zahtijevala računalnu platformu koja nije bila odmah dostupna. Mobilnom tehnologijom to se mijenja. Sad možemo na mjestu slikati tekst, obraditi i procesuirati sliku, te kao rezultat dobiti izgovor izvučenog teksta[29].

Uzbudljive nove aplikacije kao što su čitač poslovnih kartica ili prevoditelji teksta rade se za mobilne platforme. Ove aplikacije omogućuju ljudima da pomoću mobitela trenutno dobiju potrebne im informacije. Uza sve ove napretke još je uvijek relativno mali broj dostupnih **slobodnih okruženja** *(open source framework)* za mobilne uređaje[29].

Tradicionalno, mobilni uređaji su se koristili kao bežični prijenosnici zvuka. Danas je njihova svrha proširena. S dostupnošću više memorije i procesorske snage, mobilni uređaji su evoluirali iz jednostavnog komunikacijskog uređaja u snažno malo prijenosno računalo[29].

Kvaliteta senzora, na primjer kamere, ovisi o modelu mobilne platforme. U većini slučajeva fotografije sadrže **buku** *(noise),* te su nejasne zbog nedovoljno fiksirane kamere za vrijeme fotografiranja *(motion blur).* Osim toga, trenutne platforme još nisu hardverski dovoljno snažne, te ne sadrže dovoljno memorije da bi u potpunosti zadovoljavajuće izvodili kompleksne algoritme koje navedene tehnologije zahtijevaju.

Usprkos ovim zamjerkama, mobilne platforme sve više evoluiraju i sazrijevaju kao pouzdana i snažna tehnologija, te se sa sigurnošću može predvidjeti da će navedena ograničenja biti ubrzo prevladana[29].

Svrha ovog rada je opis i prikaz implementacije tehnologija OCR i TTS u kontekstu AR – a. Zbog toga će biti prikazan rad mobilne aplikacije koja objedinjuje navedene tehnologije u jednu cjelinu.

5.1 Problematika implementacije

Danas je većina mobilnih uređaja opremljena s kamerama visoke ili barem dobre rezolucije, te je moguća trenutna obrada fotografija. Takvo nešto bilo je nekad ostvarivo samo uz korištenje digitalnih kamera. Prvi mobilni OCR uređaji se pojavljuju 2001/2002 godine. Ipak, mobilna tehnologija još nije uznapredovala dovoljno daleko da omogući jednaku kvalitetu kao tradicionalna primjena. Mobilni telefoni još uvijek imaju manju količinu memorije i procesorske snage u odnosu na stolna *(desktop)* računala. Potrebna je obrada velikog broja piksela u relativno kratkom vremenu, što predstavlja problem kod fotografija s visokom rezolucijom. Dodatni problem kod mobilnih platforma je nedostatak računalne podrške kod procesuiranja realnih brojeva *(real numbers)*. Operacije obrade realnih brojeva su simulirane korištenjem **emulacijskih biblioteka** *(emulation libraries)* koje jednostavno nisu dovoljno brze, te i same opterećuju procesor. Dodatni problem predstavlja i mala veličina ekrana, zbog čega je veoma teško ostvariti efikasno sučelje[29].

Mnoge komercijalne aplikacije su dostupne na mobilnim uređajima. Veliki broj njih koristi vanjski server koji obavlja tehnološki "skupe" operacije OCR obrade. Zbog navedenih ograničenja i lakše implementacije, izvedba mobilne aplikacije koja se demonstrira primjenjivat će također vanjski server.

5.2 Dizajn sistema i specifikacije

Kao mobilnu platformu koristi se **Samsung Wave 8530** s **BADA 2.0** operacijskim sistemom.

Kao eksterni server u uporabi je **IBM NetVista** koji koristi **Ubuntu 11.10** i **Apache 2.0 HTTP server**.

Mobilna aplikacija je napisana u C++ programskom jeziku, a server strana u mješavini PHP/Linux Shell skriptnom jeziku.

Kao OCR program koristit će se Tessereact OCR.

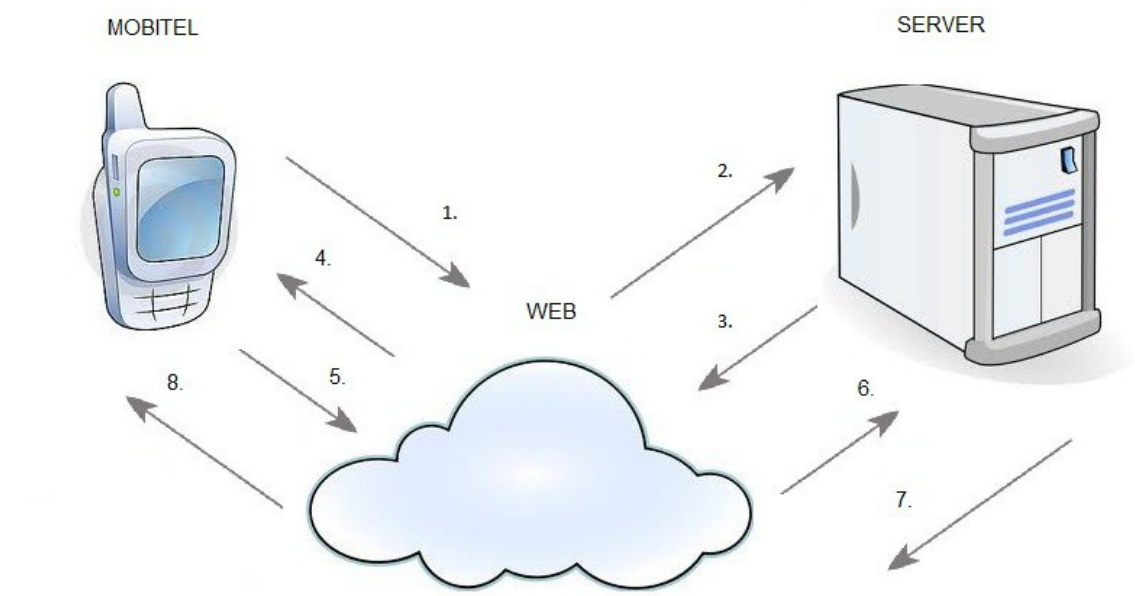
TTS je mješavina Festival TTS – a i espeak TTS – a sa MBROLA fonemima, ovisno o jeziku koji se sintezira.

5.3 Metodologija

Koristeći sisteme slobodnog koda razvijen je OCR/TTS sistem. Digitalnom kamerom na mobitelu fotografira se tekst koji želimo izlučiti. Fotografija se sačuva u rezoluciji 640x480, iako je moguće i u višoj, ali ista daje najbolje rezultate. Korisnik je tada u stanju obraditi (očistiti) fotografiju. Obrada fotografije će biti detaljno opisana kasnije, za sada je dovoljno znati da omogućuje čišćenje buke i nepotrebnih podataka koristeći filtre.

Nakon obrade fotografija se enkodira iz **JPEG** formata u **Base64** format. Dobiveni podatci se zapakiraju u **HTTP zahtjev** *(request)* paket, koji se nakon toga koristeći HGPRS ili bežičnu *(wireless)* mrežu putem interneta pošalje na server. Na serveru se dobiveni paket prvo dekodira iz Base64 u JPEG format. Dobivena fotografija se proslijedi OCR sistemu koji nakon obrade vrati tekst dobiven iz fotografije. Taj isti tekst se opet enkodira u Base64 format te se preko interneta pošalje kao **HTTP odgovor** *(response)* paket nazad na mobitel.

Na mobitelu se dobiveni tekst prvo dekodira iz Base64 u UTF8 format, da bi se nakon toga prikazao korisniku na ekranu mobitela. U slučaju da OCR nije dovoljno precizno obradio fotografiju, moguće je editirati vraćeni tekst ili čak unijeti u potpunosti novi. Navedeni tekst se opet enkodira u Base64, pošalje na server, gdje se dekodira i proslijedi TTS sistemu. TTS vrati .wav datoteku koja se radi smanjenja veličine i analogno tome kraćeg vremena prijenosa konvertira u .mp3 format. Konačni rezultat se enkodira u Base64, vrati mobitelu, gdje se opet dekodira iz Base64 u MP3 format, te je konačno spreman za slušanje.



1. Šalje su HTTP request koji sadrži Base64 enkodirani sadržaj - sliku teksta
2. Kada HTTP paket dođe do servera slika se dekodira i obradi u OCR sistemu
3. Dobiveni rezultat - tekst sa slike, se enkodira u Base64 i pošalje nazad na mobitel
4. Kad HTTP paket dođe do mobitela, dekodira se iz Base64 u UTF8 Format, te se omogući korisniku da mijenja tekst po želji
5. Ako želi, korisnik je u stanju poslati enkodirani teksti (kao HTTP paket), nazad na server, ovaj put zahtjevajući njegov zvuk
6. Kad HTTP opet paket dospije do servera dekodira se iz Base64 nazad u UTF8 te se proslijedi TTS sistemu koji vrati .wav
7. Dobiveni .wav se konvertia u .mp3, te se kao Base64 pošalje nazad na mobitel
8. Dobiveni Base64 se dekodira nazad u mp3, te je spreman za slušanje

Slika 20. Metodologija rada

5.3.1 Base64

Base64 je reverzibilna metoda enkodiranja koja konvertira 8 – bitne podatke u 7 – bitne ASCII znakove. Svaka tri bajta *(byte)* originalnih podataka se podijele u 6 – bitne blokove koje se prezentira sa četiri 7 – bitnih ASCII znakova. Ovaj proces povećava veličinu podataka za trećinu. Base64 se koristi za prijenos ne – tekstualnih podataka putem interneta, tipično kao privitak kod računalne pošte *(email attachemnt)*[30].

5.3.2 HTTP POST

U računalnoj tehnologiji POST je jedna od mnogih metoda podržanih od strane HTTP protokola. POST metoda se koristi kad klijent želi poslati podatke

serveru *(upload),* nasuprot GET metodi gdje se šalju samo URL zaglavlja *(header).* POST zahtjev sadrži i tijelo poruke *(message body),* koje može sadržavati neograničen broj podataka. Zaglavlje u POST metodi većinom samo opisuje podatke unutar tijela poruke[31].

5.4 Detaljni opis korištenih tehnologija

5.4.1 Samsung Wave 8530

Samsung Wave II 8530 (prikazan na slici 21.) je nasljednik 8500 modela. Originalno je koristio BADA 1.2 OS, ali je zbog potreba kompatibilnosti sa službenim Samsung BADA SDK unaprijeđen *(updated)* u BADA 2.0 OS. Hardverske karakteristike uključuju:

- S5PC110 CPU frekvencije 1 GHz ARM Cortex-8 CPU
- PowerVR SGX 540 grafički sustav *(graphic engine)*
- "Super LCD" ekran 720p, 480 x 800 dimenzija
- 256MB+128MB RAM – a
- Wi-Fi 802.11 b/g/n, Wi-Fi hotspot
- Kamera: 5 MP, 2592 x 1944 piksela, auto – fokus, LED flash

Mobitel je većinom napravljen od metalnih slitina, te 10.9 mm debeo. Sadrži tri navigacijske tipke: potvrdno, negativno i glavni meni[32]-

Slika 21. Samsung Wave 8530

5.4.2 BADA 2.0

Bada (koreanski – ocean) je operacijski sustav za mobilne uređaje. Razvio ga je Samsung, te predstavio u veljači 2010. Trenutna najnovija verzija je 2.0. Zanimljiva je činjenica da je BADA OS lokalizirani ugrađeni softver *(firmware)*, te se zbog toga nadograđiuje u svakoj regiji zasebno. Zbog te činjenice nadogradnje se ne izvršavaju u svim regijama u isto vrijeme, već često kasne. Službena Bada 2.0 još nije izašla u Hrvatskoj regiji (BiH, SLO i CRO), dok je u Mađarskoj regiji izašla prije skoro 8 mjeseci. Također je zanimljivo da službeni SDK za razvoj Bada OS aplikacija, podržava samo Bada 2.0. Zbog toga veliki broj programera iz različitih regija nije u stanju testirati svoje aplikacije direktno na mobitelima svoje regije, već moraju različitim hakerskim *(hacker)* metodama instalirati druge regionalne ugrađene softvere ili koristiti simulator[33].

Bada je platforma bazirana na Linux jezgri *(kernel).* Koristi kod od FreeBSD, NetBSD i OpenBSD Linux OS – a.

Bada korisniku omogućava korištenje različitih UI kontrola. Dopušta osnovne UI kontrole kao *Listbox, Color Picker i Tab.* Podržava Adobe Flash 9 (1.2 Bada) i 10 (2.0 Bada) i *OpenGL ES 2.0* 3D Grafički API *(3D graphics AP)*.

Bada aplikacije se razvijaju u C++ programskom jeziku koristeći Eclipse bazirani IDE *(integrated development environment).* Za testiranje i pronalaženje grešaka *(debug)* postoji emulator koji je u stanju pokretati aplikacije.

5.4.3 IBM NetVista

Da bi se ubrzao rad OCR/TTS dijela aplikacije, tj. da bi se mobilna aplikacija oslobodila izvođenja skupih izračuna, koristi se server uređaj. Pošto se demonstrirana aplikacija ne koristi od strane velikog broja korisnika, opterećenje na server neće biti visoko. Zato se bez problema može koristi uređaj sa slabijim performansama, u ovom slučaju to će biti IBM NetVista računalo. NetVista IBM serija je okarakterizirana dugovječnim rokom trajanja komponenti i osrednjim radnim performansama. Model koji se primjenjiva ovom slučaju je NetVista P-IV 1.6GHz.

Hardverske komponente [34]:

- 1.6GHz Pentium 4 Systems
- 256MB PC133 SDRAM
- 40GB HDD
- AGP Video Card
- 10/100Mbps LAN
- CDROM/USB/FLOPPY

5.4.4 UBUNTU 11.10

Operacijski sustav koji pokreće server je **Linux Ubuntu**. Ubuntu je baziran na Debian Linux distribuciji kao slobodan i otvoren softver. Ubuntu je sponzoriran od strane Britanske tvrtke *Canonical Ltd.* koja je u vlasništvu Južno – Afričkog poduzetnika Marka Shuttelwortha. Riječ Ubuntu je Južno – Afričkog podrijetla, te označava filozofiju **Ubuntu** – humanizam prema drugima. Ubuntu OS se primarno koristi na osobnim *(desktop)* računalima, ali i kao server kod početnika. Odlikuje se odličnom dokumentacijom (za jedan Linux) i širokom zajednicom korisnika.

Ubuntu je prvi put izdan 20. listopada 2004. Od tada nova verzija izlazi svakih 6 mjeseci. Trenutna verzija je 11.10. Ubuntu paketi su bazirani na **Debian grani nestabilnih paketa** *(Debian's unstable branch)*, obje distribucije koriste **.deb** paket format. Debian i Ubuntu paketi nisu striktno binarno kompatibilni, ponekad .deb paketi moraju biti prevedeni iz koda *(compiled from source)* u Ubuntu okruženju.

Ubuntu se sastoji od mnogo **softver paketa** *(software packages)* od kojih je većina distribuirana pod slobodnom licencom. Jedina iznimka su određeni zakonom zaštićeni *(proprietary)* hardverski upravljački programi *(drivers)*.

Razlog zbog kojeg je odabran Ubuntu je činjenica da je besplatan i da sadrži veliki broj besplatnih programa (kao OCR i TTS) koji se mogu skinuti sa Ubuntu repozitorija *(repository)*[35].

5.4.5 APACHE 2.0

Apache HTTP web server (još poznat kao Apache) je web server koji od 2009 pokreće više od 100 milijuna web stranica. Trenutno je najpopularniji web server na kojem je 57.46% svih Internet stranica. Apache se prvi put pojavio kao alternativa *Netscape Communications Corporation* web serveru, te je od tada sazrio u ozbiljan i brz web server. Apache se tipično pokreće na Unix/Linux operativnim sistemima.

Kod rada sa mobilnom aplikacijom, web server je potreban da bi omogućio komunikaciju preko interneta[36].

5.4.6 Tessereact OCR

Tesseract je slobodni, više – platformni *(cross - platform)* softver koji se koristi za znakovno prepoznavanje. Originalno je razvijan između 1985 i 1995 od strane Hewlett-Packard kao komercijalni softver. 2005 je otvoren kao softver slobodnog koda. Od 2006 Google sponzorira njegovo daljnje razvijanje.

Tesseract je 1995 bio jedan od 3 najbolja OCR sistema. Dostupan je na platformama kao Linux, Windows i Mac OS, ipak zbog limitiranih resursa detaljno se testira jedino na Windows i Ubuntu Linux.

Inicijalni Tesseract je bio u stanju prepoznavati samo engleski tekst, dok novije verzije su u stanju prepoznavati veliki broj jezika kao hrvatski, grčki, španjolski itd..[37]

5.4.9 MBROLA

MBROLA je algoritam za sintezu govora. Distribuira se besplatno, ali u binarnom formatu. MBROLA projekt dobavlja bazu podataka fonema za veliki broj jezika. MBROLA softver nije potpuni TTS sistem, tekst mora prvo biti transformiran u informacijski MBROLA format koji opisuje foneme i prozodiju. Kvaliteta MBROLA sinteze je veća od većine fonetskih sintetizatora. Glavni razlog tome je činjenica da je bazirana na pre – procesuiranju fonema što dosta poboljšava njihovu konkatenaciju dok malo smanjuje kvalitetu segmenata. MBROLA je globalni projekt sa rekordnim brojem individua iz različitih zemalja koji su sudjelovali u njegovoj kreaciji.[38]

5.4.8 Festival TTS

Festival TTS je višejezični sistem sinteze govora, originalno razvijen od strane Alan W. Blacka u **Centru za istraživanje govorne tehnologije** *(Centre for Speech Technology Research (CSTR))* na Sveučilištu u Edinburgu. Koristi metodu konkatinacije govornih jedninica. Distribuira se kao slobodni softver. Festival nudi puni TTS API i IDE za razvijanje i istraživanje sinteze teksta. Napisan je u C++ i dizajniran tako da podržava veliki broj jezika.[39]

5.4.9 eSpeak

Espeak TTS je slobodni softver sinteze govora. Dostupan je na Linux, Windows i Mac platformi. eSpeak je moguće koristiti u kombinaciji sa MBROLA fonemima. Koristi se i kao baza TTS – a Google prevoditelja *(Google Translate).*

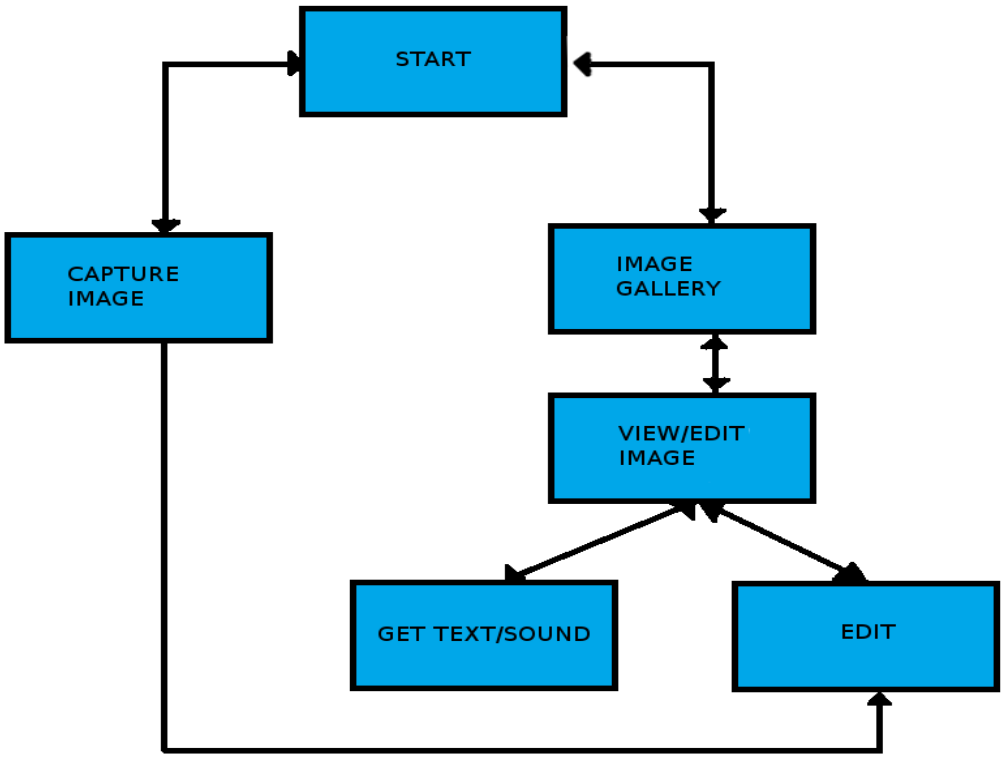
5.5 Klijent – Mobilna aplikacija

Cilj mobilne aplikacije je pružiti korisniku uređaj koji će moći izlučiti tekst s fotografije. Da bi se to postiglo mobilna aplikacija mora biti u stanju:

- Fotografirati tekst
- Osigurati zadovoljavajuću kvalitetu fotografije
- Omogućiti pospremanje fotografije
- Omogućiti prikazivanje i editiranje fotografije u svrhu čišćenja buke
- Ostvariti konekciju sa serverom
- Konvertirati sliku u odgovarajući format
- Slati podatke na server
- Primiti i obraditi dobivene podatke sa servera
- Igrati zvuk

5.5.1 Korisničko sučelje

Domaći *(native)* programski jezik Bada aplikacija je C++. Korisničko sučelje *(user interface - UI)* podržava standardne UI komponente kao *textBox, listBox* itd. Dijagram toka UI je prikazan na slici 22. Pri demonstraciji rada aplikacije koristit ćemo slike ekrana *(screenshot)* aplikacije pri radu u Bada IDE simulatoru.



Slika 22. Dijagram toka UI

5.5.2 Start – početni ekran

Početni ekran prikazan na slici 23. pojavljuje se nakon što je korisnik pokrenuo aplikaciju. Korisnik je u stanju odlučiti da li želi raditi s već otprije snimljenom fotografijom ili snimiti novu. Navigacijska kontrola **gumb** *(button)* je prikazana s dvije slike i tekstualnim opisom. Pritiskom na kontrolni gumb prelazi se na drugi ekran u aplikaciji.

Lijeva slika prikazuje malu kameru i engleski natpis *Capture image.* Pritiskom

na istu prelazi se na ekran snimanja nove slike.

Desna slika prikazuje olovku i papir, što bi trebalo asocirati korisnika na editiranje nekog sadržaja, u ovom slučaju slike. Pritiskom na istu prelazi se na ekran galerije slika.

Slika 23. Početni Izbornik

5.5.3 Galerija slika

Pretpostavit ćemo da je korisnik pritisnuo **View/Edit Image** sliku na početnom ekranu, tj. da je odlučio editirati već otprije snimljenu sliku. Prelaskom na ekran galerije (prikazan na slici 24.) učitavaju se sve fotografije iz */Images/Camera* direktorija te se unose u *Bada UI List* komponentu.

Slika 24. Galerija slika

Bada List komponenta je u stanju prikazati tekst i sliku na ekranu mobitela, u ovom slučaju prikazivat će se samo fotografije. Proslijeđene fotografije se automatski skaliraju u manju veličinu tako da ih se prikaže po tri u redu.

5.5.4 Ekran prikaza fotografije

Odabirom fotografije (pritiskom na istu) stvara se nova instanca **Ekrana prikaza fotografije** *(View/Edit)*. Nakon toga mu se prosljeđuje putanja odabrane fotografije, da bi se na kraju ekran prikaza pojavio s odabranom slikom na pozadini, kao što se može vidjeti na slici 25.

Slika 25. Ekran prikaza fotografije

U dnu slike 25. možemo vidjeti dva gumba s natpisima *(label)* "Get Text" i "Edit", te navigacijski gumb za povratak na prijašnji ekran.

5.5.5 Ekran editiranja

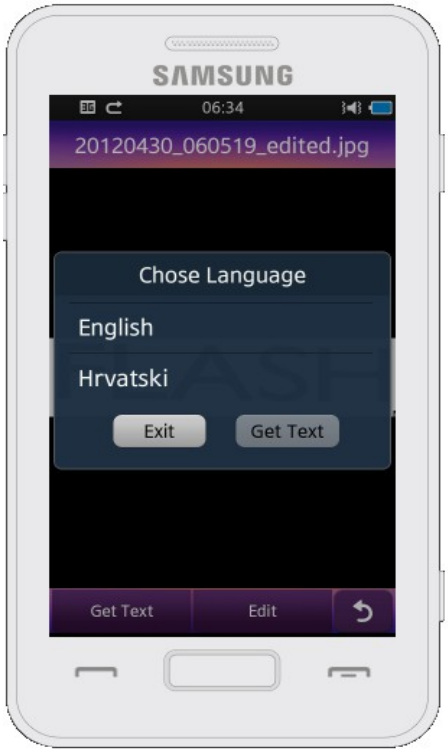
Nakon što je odabrana željena fotografija, korisnik treba odluči dali želi poslati fotografiju na server da bi se iz nje izlučio tekst ili dodatno obraditi fotografiju čime bi se smanjila količina buke te olakšao posao OCR sistemu. Na slici 25. vidi se da je fotografija loše orijentirana i da sadrži previše sadržaja. Zbog toga bi bilo veoma poželjno obraditi je. Pritiskom na gumb "Edit" prelazi se na ekran editiranja fotografije. Proces editiranja će biti detaljno opisan u poglavlju 5. Za sada ćemo na slici 26. prikazati konačni rezultat obrade.

Slika 26. Rezultat obrade

Na slici 26. vidi se da je odabran samo onaj tekst koji se želi transformirati u zvuk, te da je očišćena sva buka s njega. Fotografija je također monokromatska, tj. striktno crno – bijela, čime smo uvelike olakšali rad OCR -u.

5.5.6 Slanje fotografije na server i obrada

Povratkom na ekran prikaza ponovno dobivamo izbor editiranja ili slanja fotografije na server. Pošto je sad fotografija obrađena treba je poslati na server, nakon čega se nazad dobiva tekst. Pritiskom na "Get Text" gumb otvara se novi dijalog koji nam daje izbor odabira jezika teksta.



Slika 27. Dijalog odabira jezika

Nakon odabira jezika fotografija se zapakira u **HTTP POST** paket, te se pošalje na *http://macetini.mooo.com/upload/text/index.php* URL. Domena *macetini.mooo.com* je ustvari pod – domena *(subdomain)* besplatne domene mooo.com. Koristeći besplatni DNS klijent *(dns client – inadyn)* registrira se IP adresa rutera *(router)* na koji se namjenski *(dedicated)* server spaja. Sav HTTP (port 80) promet koji stigne na tu adresu se proslijedi *(port forwarding)* namjenskom serveru, te shodno tome i APACHE web serveru. Koristeći PHP skriptni jezik prvo izlučimo sliku iz POST paketa, te joj promijenimo format iz JPEG u TIFF. To učinimo iz tog razloga što tesseract OCR može raditi samo s TIFF formatom. Nakon konverzije pozivamo sistemsku skriptu koja kao ulazni argument uzima dobivenu fotografiju i prosljeđuje je tesseract OCR – u. Kad se OCR obrada završi, dobiveni tekst se enkodira u Base64 format i šalje kao HTTP odgovor nazad.

4.5.7 Prikaz rezultantnog teksta i slanje istog na server

Kad HTTP odgovor stigne na mobitel dekodira se nazad u UTF format i

prikaže na ekranu unutar Bada *UI EditArea*[41], kao što je prikazano na slici 28. Može se primijetiti da je u donjem desnome kutu gumb "Play" onemogućen *(disabled).* Razlog tome je taj da još nismo poslali tekst na server koji bi nam u odgovor trebao vratit govor istog.

Slika 28. Rezultat obrade

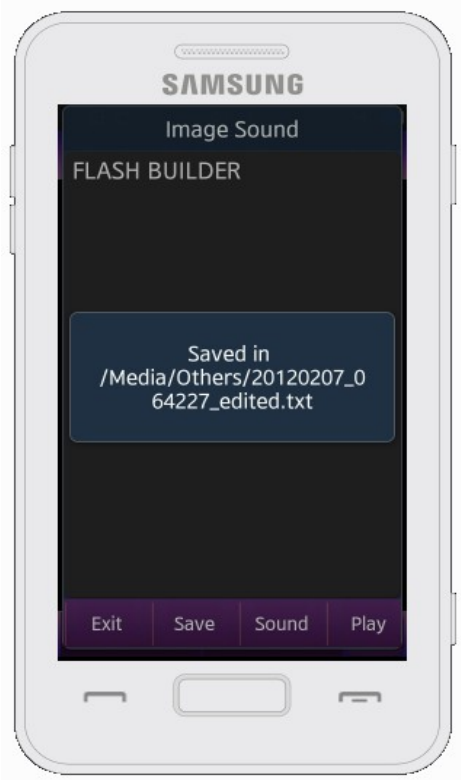
Korisnik je sad u stanju mijenjati tekst ili unijeti sasvim novi. Editiranje teksta se inicira pritiskom bilo gdje unutar UI EditArea komponente, čime ona prelazi u **mod obrade** *(edit mode).* Primjer obrade teksta je prikazan na slici 29.

Slika 29. Obrada teksta

Kad je tekst zadovoljavajuće obrađen, pritisnemo gumb "Sound" koji pokreće proces dobavljanja govornog zvuka. Tekst i odabrani jezik se opet šalju kao POST HTTP paket na server, ali ovaj put na drugu URL adresu: *http://macetini.mooo.com/upload/sound/index.php* . Na serveru se iz HTTP paketa izluči Base64 tekst i jezik istog, te ih se dekodira u UTF format. Na temelju argumenta jezika odlučuje se koji TTS će se koristit za sintezu govora. Festival daje kvalitetniji govor kod engleskog jezika, a eSpeak u kombinaciji s MBROLA projektom daje zadovoljavajući hrvatski govor. Jezik se proslijedi sistemskoj skripti koja pozove odgovarajući TTS. Dobiveni rezultantni zvuk dolazi u WAV formatu. Da bi se smanjila veličina datoteke, navedena skripta konvertira dobiveni rezultat u MP3 datoteku koja je ponekad deset puta manja od originalne WAV datoteke. MP3 datoteka se enkodira u Base64 format, te šalje kao HTTP odgovor nazad na mobilnu platformu. Kad stigne nazad dobiveni Base64 se dekodira u MP3, te je spreman za slušanje, koje se odvija pritiskom na sada omogućeni *(enbled)* "Play" gumb.

4.5.8 Dodatna funkcionalnost – spremanje teksta i zvuka

U bilo koje vrijeme moguće je lokalno spremiti dobivene podatke. Pritiskom na gumb označen labelom "Save" tekst vidljiv na ekranu spremi se kao datoteka s .txt ekstenzijom unutar */Media/Others* direktorija na mobitelu. Primjer spremanja je vidljiv na slici 30. Ako je unutar međuspremnika *(buffer)* memorije prisutna MP3 datoteka, ona se spremi u */Media/Sounds* direktorij. Spremljenim podatcima se može pristupiti kad god želimo koristeći Bada datotečni sustav *(file system).* Imena datoteka su preslike naziva fotografija, koji su ustvari sekvencijalni brojevi, npr. slika 0000023.png se preslika u 0000023_edited.txt i 0000023_edited.mp3.



Slika 30. Čuvanje podataka

5.5.9 Slikanje teksta

Na početnom izborniku osim obrade već postojećih fotografija, moguće je i snimiti nove. Pritiskom na navigacijsku tipku označenu s "Capture image" pokreće se kamera i pokreće korisnički upravljački servis prikazan na slici 31.

Slika 31. Korisnički upravljački servis

Fotografija se sad može snimiti pritiskom na gumb označen sa slikom male kamere. Osim što se može slikati, moguće je podešavati specifikacije kamere npr. kontrast, bljesak, fokus itd... Nakon što je fotografija snimljena, mobilna aplikacija je pohranjuje u */Media/Images* direktorij, te prelazi na ekran obrade fotografije gdje je moguća njena direktna obrada.

6 Obrada fotografija

Pri radu sa fotografijama velika količina pozadinskog sadržaja nepotrebno opterećuje OCR sustav. Zbog toga mobilna aplikacija omogućuje korisniku obradu fotografija, pomoću čega se eliminira pozadinska buka i povećava kvaliteta sadržaja koji se šalje na server. Fotografija se obrađuju koristeći različite **filtre obrade**. Filtar obrade je algoritam koji kao ulazni parametar funkcije prima neku fotografiju, te na temelju određenih pravila propušta ili mijenja poziciju i svojstva piksela navedene fotografije.

Filtre obrade možemo podijeliti na:

- transformacijske filtre – koji mijenjaju rotaciju, direkciju itd..
- filtre svojstva – koji mijenjaju svojstva piksela fotografije, kao RGB vrijednosti

Filtrima obrade pristupamo prelaskom s ekrana **Prikaza fotografije** na ekran **Editiranja** *(Edit)*. Na slici 32 je primjer ekrana editiranja. Fotografija koja se editira je prikazana u pozadini aplikacije. U vrhu aplikacije imamo dva gumba **Adjust** i **Transform**. "Adjust" gumb sadrži filtre svojstva, a "Transform" transofrmacijske filtre.

Slika 32. Ekran editiranja

Kao što možemo vidjeti fotografija je zagušena s velikom količinom nepotrebnih podataka. Ako se pretpostavi da korisnik želi izlučiti slovo C s fotografije i transformirati ga u OCR razumljiv oblik, onda se moraju izvesti sljedeće radnje:

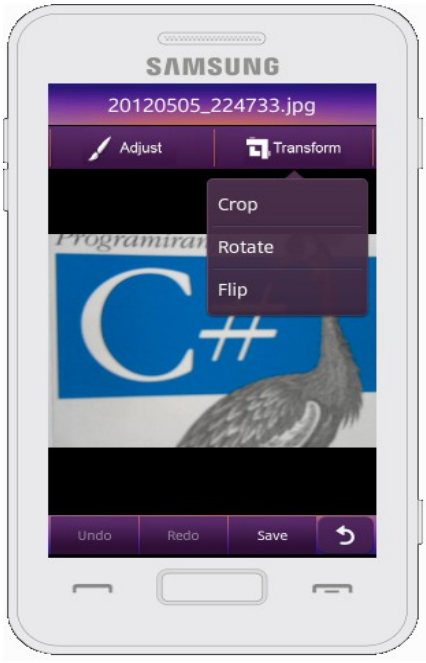
1. Rezanje samo potrebnog dijela fotografije
2. Konvertiranje fotografije u oblik **sive skale** *(grayscale)*
3. Korištenjem algoritma praga očistiti svu buku s fotografije

6.1 Transformacijski filtri

Pritiskom na "Transform" gumb otvara se padajući izbornik *(dropdown meni)* prikazan na slici 33. Unutar izbornika se mogu vidjeti tri opcije:

- Crop
- Rotate
- Flip

Pritiskom na bilo koju od njih otvara se sučelje odabranog filtra.



Slika 33. Padajući izbornik

6.1.1 Crop filtar

Većinom se pri fotografiranju sačuva velika količina nepotrebnog, što opterećuje OCR i povećava veličinu fotografije. Na slici 32. želimo izlučiti samo slovo C, zbog toga je implementiran **filtar rezanja** *(Crop).*

Pritiskom na "Crop" opciju unutar "Transform" padajućeg izbornika na ekranu mobilne aplikacije, prikazuje se kvadrat čije je dimenzije moguće mijenjati pomicanjem sjecišta pravaca njegovih strana. Na slici 34. vidi se zaokruženi dio fotografije koji je potrebno izrezati.

Slika 34. Željeni tekst

Odabirom kvačice na dnu, svi pikseli unutar kvadrata se preslikaju kao nova slika, čime je efektivno odabran samo potreban dio. Na slici 35. je prikazan rezultat Crop fitra.

Slika 35. Rezultat Crop Filtra

6.1.2 Rotate filtar

Fotografiranje nekog prostora ne daje uvijek idealne rezultate. Ponekad je zbog dimenzijskih ograničenja mobitela potrebno željeni sadržaj fotografirati pod kutem, što može zbuniti OCR. Tijekom razvoja mobilne aplikacije predviđeno je ovo ograničenje, zbog toga je moguće sliku rotirati za 90° u bilo kojem smjeru. Isto tako omogućeno je u potpunosti okrenuti direkciju *(flip)* fotografije. Funkcionalnosti orijentacije i okretanja su izvedene bez korisnički prilagođenih algoritama, već korištenjem službenih BADA sistemski **ImageUtil** funkcija.

Odabirom "Rotate" gumba unutar padajućeg izbornika prikazuje se sučelje Rotate filtra. Na dnu slike 38. vide se dvije navigacijske slike koje označavaju smjer željene rotacije i sliku zelene kvačice koja predstavlja završetak editiranja.

Slika 36. Rotacijski filtar

Pritiskom na rotacijske gumbe fotografija se rotira za 90° u bilo kojem smjeru. Kad je fotografija pod potrebnim kutem, pritisne se kvačica, te je proces gotov. Konačni rezultat je prikazan na slici 37.

Slika 37. Rezultat obrade filtrom

6.1.3 Flip Filtar

Filtar promjene orijentacije je implementiran u slučaju da je snimljena fotografija iz nekog razloga orijentirana na krivu stranu, kao na primjer odraz u ogledalu. Razlog ove implementacije je činjenica da samom rotacijom nije moguće ispraviti takav defekt. Odabirom "Flip" gumba iz padajućeg izbornika prikazuje se sučelje Flip filtra. Na slici 38. vidi se primjer krivo orijentirane fotografije i Flip filtar sučelje.

Slika 38. Krivo orijentirana fotografija i sučelje flip filtra

Pritiskom na sliku strijele mijenja se direkcija fotografije, tj. okreće se u smjeru navedene strijele.

6.2 Filtri svojstva

Do sada su opisani transformacijski filtri. Za razliku od navedenih, filtri svojstva se bave svojstvima piksela, tj. njihovim RGB vrijednostima. Na ekranu editiranja prikazanog na slici 39. pritiskom na gumb "Adjust" otvara se padajući meni s opcijama:

- Grayscale

- Threshold
- Flip color
- Clear

Slika 39. Filtri svojstva

Pritiskom na bilo koji od ovih gumba primjenjiva se filtar ili se pokreće njegovo korisničko sučelje.

6.2.1 Grayscale filtar

Grayscale filtar (Filtar sive skale) konvertira vrijednosti svih piksela na ekranu u odgovarajće vrijednosti sive nijanse. Grayscale vrijednosti piksela olakšavaju rad OCR sustavu. Pikseli se odabiru pomoću dvije **for** petlje (jedna za *x,* druga za *y.)*. Odabirom "Grayscale" gumba na ekranu automatski se primjenjiva navedeni filtar.

Algoritam konverzije u sivu skalu radi tako da se RGB vrijednosti svakog zasebnog piksela pomnože s odgovarajućim konstantama[42]:

- Crvena *(R – Red)* vrijednost se pomnoži s 0.2989
- Zelena *(G – Green)* vrijednost se pomnoži s 0.5870

- Plava (*B – Blue)* vrijednost se pomnoži s 0.1140

Na slici 40. Prikazan je prije i poslije efekt GrayScale filtra.

Slika 40. Grayscale filtar – a)prije b)poslije

6.2.2 Threshold filtar

Threshold filtar (filtar praga) je možda najvažniji filtar u aplikaciji. Njegova funkcija je da pročistiti fotografiju od buke, te dodatno odvoji tekst od pozadine. U opisu rada OCR sistema, veliki dio je posvećen pronalaženju optimalnog filtra praga. U mobilnoj aplikaciji koja se demonstrira, korisnik je u stanju prije same OCR obrade primijeniti svoj filtar praga.

Prag predstavlja RGB vrijednost pomoću koje se stvaraju binarne fotografije *(binary images). S*vaki piksel koji ima RGB vrijednost veću od navedenog praga poprima RGB vrijednost 255, 255, 255 tj. crnu, odnosno 0, 0, 0 tj. bijelu u slučaju da je navedena vrijednost manja[43].

Pritiskom na "Threshold" gumb pokreće se sučelje navedenog filtra. Sučelje se sastoji od klizača *(slider),* plus, minus, potvrdna i negativna kontrola. Pomicanjem

klizača ili kontrola povećava se ili smanjuje vrijednost praga. Pri obradi, korisnik je u stanju trenutno mijenjati iznos praga i vidjeti fotografiju nakon promjene istog. Primjer uporabe filtra vidi se na slici 41.

Slika 41. Uporaba Threshold praga

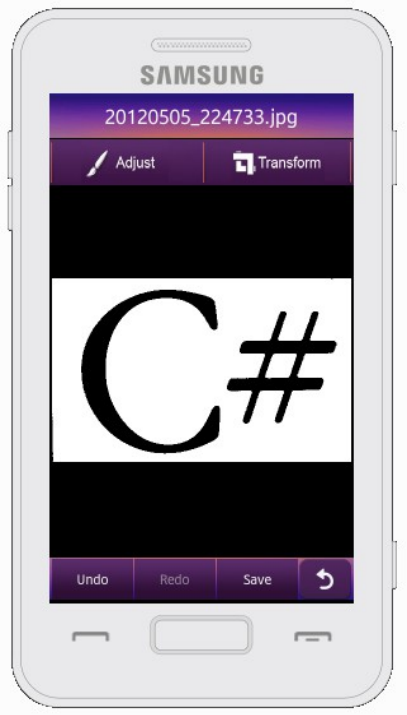
Zbroj RGB vrijednosti piksela maksimalno može iznositi 765 (3*255). Klizač praga je ustvari vrijednost od 0 do 255, koju kad mijenjamo primjenjivamo novi filtar praga. Zbroj RGB vrijednosti svakog piksela se uspoređuje s trenutnom pozicijom klizača pomnoženom s tri. Ako je zbroj RGB vrijednosti trenutnog piksela manji od vrijednosti na klizaču, RGB vrijednosti navedenog piksela se postavljaju na 0, 0, 0, tj. bijelu (pozadina), a ako je veća onda se postavljaju na 255, 255, 255 tj. crnu (tekst).

6.2.3 Flip Color Filtar

Flip Color **(Izokreni boje)** je jednostavan filtar koji je u stanju izokrenuti boje fotografije – crno postaje bijelo, bijelo postaje crno. Razlog njegove implementacije je moguće povećavanje preciznosti OCR sustava. Tesseract OCR daje najbolje rezultate kod crnih slova s bijelom podlogom.

Odabirom na Flip Color iz padajućeg izbornika pokreće se funkcionalnost

navedenog filtra. Svaki piksel koji ima zbroj RGB vrijednosti 765 postavlja se na 0 (RGB – 0,0,0), a koji ima RGB zbroj 0 postavlja se na 765 (RGB, 255,255,255). Zbog navedene funkcionalnosti Flip color filtar je najbolje koristiti nakon primjene Threshoald filtra. Na slici 42. vidi se primjena Flip color filtra.

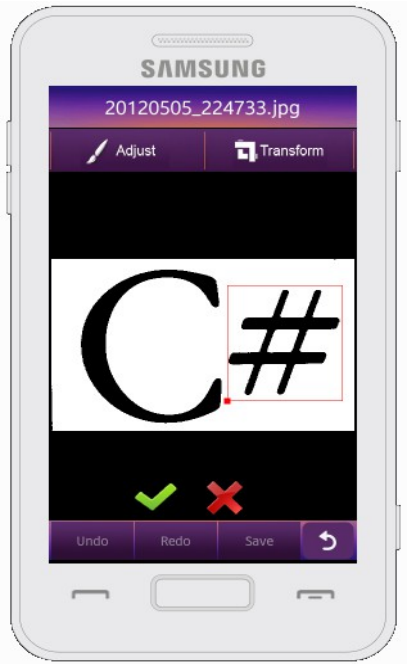


Slika 42. Primjena Flip color filtar

6.2.4 Clear Filter

Clear "Očisti" filtar, se koristi kako bi se izbrisali nepotrebni podatci s fotografije.

Kao i kod "Crop" filtra odabirom na "Clear" opciju unutar "Adjust" padajućeg izbornika na ekranu mobilne aplikacije prikazuje se kvadrat čije je dimenzije moguće mijenjati pomicanjem sjecišta pravaca njegovih strana. Na slici 43. vidi se zaokruženi dio fotografije koji je potrebno očistiti.



Slika 43. Tekst koji je potrebno izbrisati

Odabirom kvačice na dnu, svi pikseli unutar kvadrata se postavljaju na vrijednost 0,0,0 tj. bijelu čime su efektivno izbrisani nepotrebni podatci. Na slici 44. je prikazan rezultat Crop filtra.

Slika 44. Rezultat upotrebe "Clear" filtra

6.2.5 Undo/Redo/Save

Pri radu s aplikacijom ponekad se dogodi da korisnik želi poništiti promjene, tj. vratiti se jedan korak nazad ili čak vratiti nazad poništeno – korak naprijed. Pritiskom na "Undo" *(poništi)* gumb, poništava se obrada posljednjeg filtra, a ako nakon toga nije izvedena nikakva promjena, pritiskom na "Redo" *(obnovi)* gumb, te iste promjene se vraćaju.

Ako je korisnik završio obradu i želi sačuvati svoje promjene, pritiskom na gumb "save", edititrana fotografija se sačuva unutar */Media/Images/* direktorija. Ime nove fotografije se sastavi od originalnog imena sa *edited* sufiksom. Fotografija xxxxx.png postaje xxxxx_edited.png.

7 Server strana

Do sada je opisana logika klijentskog dijela tj. mobilne aplikacije. Da bi se dobio potpuni uvid u rad aplikacije potrebno je opisati server stranu, koja je odgovorna za konverziju sadržaja dobivene fotografije u tekst i zvuk.

Server OS, u ovom slučaju Linux, je spojen na ruter *(Samsung Gigaset SX 763)* koji ima izlaz na Internet. Koristeći protokol **Prosljeđivanja ulaza** *(port forwarding)* sav HTTP promet što dođe na ruter se proslijedi IP adresi servera.

Koristeći *Inadyn*[44] DDNS *(dynamic domain name system)* klijent, server periodički izvještava *freeDNS*[45] DDNS server o promjeni IP adrese rutera čime smo omogućili njegovu dostupnost preko interneta, a analogno tome i servera.

FreeDNS server mapira IP adresu rutera s domenom *macetini.mooo.com,* te periodički izvještava ostale DNS servere o njezinom postojanju.

7.1 Linux naredbe server strane

Pri obradi fotografije Apache web server koristi PHP skripte da pristupi sljedećim Linux naredbama/skriptama:

- convert
- tesseract
- text2wave
- lame
- base64
- espeak

7.1.1 Convert (pretvorba) naredba

Naredba Pretvorbe[46] konvertira format fotografije koja dođe na server u format koji je tesseract OCR u stanju učitati. Fotografije dolaze u JPEG formatu, a

OCR koristi meta – podatcima bogat TIFF[47] format. Primjer poziva naredbe s argumentima se vidi na primjeru 1.

convert img.jpg img.tif

Primjer 1. Poziv convert naredbe

Lista argumenata:

- img.jpg – fotografija koja se konvertira,
- img.tif – naziv i format slike u koju se konvertira

7.1.2 Tesseract naredba

Tesseract[48] naredba se koristi za pozivanje prevedene binarne datoteke tesseract OCR – a. Primjer poziva naredbe se vidi na primjeru 2.

tesseract img.tif img -l hrv

Primjer 2. Poziv tesseract naredbe

Lista argumenata:

- img.tif – slika koja se konvertira
- img – naziv slike u koju se konvertira
- -l hrv – opcija i ime jezika teksta

7.1.3 text2Wave skripta

TextToWave[49] je Festival TTS skripta koja kao ulazne argumente prima TXT datoteku čiji tekst kovertira u WAV zvučnu datoteku. Primjer poziva skripte se vidi na primjeru 3.

text2wave img.txt -eval '(voice_us1_mbrola)' -o img.wav

Primjer 3. poziv textToWave skripte

Lista argumenata:

- img.txt – ulazna datoteka koja sadrži tekst koji se sintetizira u govor.
- -eval '(voice_us1_mbrola)' – MBROLA fonemi koji se koriste pri sintezi
- -o img.wav – naziv WAV datoteke

7.1.4 Lame naredba

Lame[50] naredba konvertira WAV u MP3 datoteke. Razlog pozivanja je činjenica da je MP3 daleko manja datoteka od WAV, te je vrijeme prijenosa iste nazad na server kraće. Primjer poziva naredbe se vidi na primjeru 4.

lame img.wav img.mp3

Primjer 4. Poziv lame naredbe

Lista argumenata:

- img.wav – WAV datoteka koju je potrebno konvertirati
- img.mp3 – MP3 datoteka u koju se konvertira

7.1.5 Base64 naredba

Prijenos podataka kao fotografije ili zvučne datoteke putem interneta je najlakše izvesti kao znakovni niz. Base64[51] naredba konvertira podatke u navedeni niz. Primjer poziva naredbe se vidi na primjeru 5.

base64 img.mp3 > imgBase64.txt

Primjer 5. Poziv Base64 naredbe

Lista argumenata:

img.mp3 – ulazna datoteka, u ovome slučaju MP3

imgBase64.txt – izlazna TXT datoteka, koja sadrži base64 niz

7.1.6 eSpeak naredba

Espeak[52] naredba konvertira znakovni tekst u zvučni govor. Prosljeđivanjem dodatnih argumenata, rezultat obrade eSpeak TTS sinteze se može pohraniti u WAV datoteku. Primjer poziva naredbe se vidi na primjeru 6.

espeak -f img.txt -w img.wav -v mb-cr1 -s 125

Primjer 6. Poziv eSpeak naredbe

Lista argumenata:

-f img.txt – Tekstualna datoteka čiji se sadržaj sintetitzira u govor

-w img.wav – Zvučna datoteka koja se dobije kao rezultat obrade

-v mb_cr1 – MBROLA fonemi

-s 125 – brzina izgovora (broj riječi po minuti), u ovome slučaju 125 riječi

7.2 OCR/TTS funkcionalnost

7.2.1 OCR

Na URL adresi *macetini.mooo.com/upload/text/* nalazi se PHP *index.php* skripta koja iz HTTP POST paketa prima fotografiju i jezik u koji se tekst sa fotografije konvertira. Indeks.php skripta poziva Linux shell *prep_text.sh* skriptu koja pokreće gore navedene naredbe. Primjer 7. i 8. prikazuju rad PHP/Shell skripte:

<?php

//pomoćni kod - debuging

```
file_put_contents('./help/headers.txt', var_export(apache_request_headers(), true));

file_put_contents('./help/post.txt', var_export($_POST, true));

file_put_contents('./help/files.txt', var_export($_FILES, true));


//prebacivanje dobivene fotografije u img direktorij

move_uploaded_file($_FILES['photoFile']['tmp_name'], '../img/img.jpg');


//isčitavanje jezika

$lang = $_POST['lang'];


//poziv prep_text.sh skripte

$output =  system('./prep_text.sh '.$lang);


//isčitavanje teksta

$original=file_get_contents('../img/img.txt');


//oznaka koji tip podataka se vraća: zvuk ili tekst

echo "_TEXT__";


//ispis dobivenih podataka

echo base64_encode($original);


?>
```

Primjer 7. Indeks.php skripta

```
convert ../img/img.jpg ../img/img.tif

tesseract ../img/img.tif ../img/img -l $1
```

Primjer 8. prep_text.sh

7.2.2 TTS

Na URL adresi *macetini.mooo.com/upload/sound/* nalazi se PHP *index.php* skripta koja iz HTTP POST paketa prima tekst koji se sintetizira u zvuk i njegov jezik. Na temelju jezika u koji se konvertira, Indeks.php poziva Linux shell skriptu *prep_sound_eng.sh* ili *prep_sound_hrv.sh* koja poziva gore navedene naredbe. Primjer 9, 10 i 11. prikazuju rad PHP/Shell skripte:

```
<?php


//debugging

file_put_contents('./help/headers.txt', var_export(apache_request_headers(), true));

file_put_contents('./help/post.txt', var_export($_POST, true));

file_put_contents('./help/files.txt', var_export($_FILES, true));


//izlučivanje tijela texsta iz HTTP POST paketa

$bodyBase64 = $_POST['body'];


//izlučivanje jezika iz HTTP POST paketa

$lang = $_POST['lang'];
```

```
//dekodiranje teksta iz base64 formata
$body = base64_decode($bodyBase64);

//spremanje dobivenog teksta u img.txt datodeku
file_put_contents('../img/img.txt', $body);

//trenutni direktorij
$dir = dirname(__FILE__);

//na temelju dobivenog jezika određiva se koja skripta se poziva
if($lang == "eng")
	$system_command = "$dir/prep_sound_eng.sh ";
else
	$system_command = "$dir/prep_sound_hrv.sh ";

//poziv skripte
$output = shell_exec($system_command);

//spremanje dobivenog sadrzaja
$original = file_get_contents('../img/imgBase64.txt');

//velicina podataka sto se salji nazad na zada na mobilnu aplikaciju
```

```
$len = filesize('../img/imgBase64.txt')+7;

//spremanje veličine podataka u header, (funkcionalnost mobilne aplikacije)
header('Content-Length: ' . $len);

//oznaka vrste podataka koji se šalju
echo "_SOUND_";

//ispis podataka (AJAX)
echo $original;

?>
```

Primjer 9. /sound/Index.php skripta

```
text2wave ../img/img.txt -eval '(voice_us1_mbrola)' -o ../img/img.wav
lame ../img/img.wav ../img/img.mp3
base64 ../img/img.mp3 > ../img/imgBase64.txt
prep_sound_hrv.sh
```

Primjer 10. prep_sound_eng.sh skripta

```
espeak -f ../img/img.txt -w ../img/img.wav -v mb-cr1 -s 125
lame ../img/img.wav ../img/img.mp3
base64 ../img/img.mp3 > ../img/imgBase64.txt
```

Primjer 11. prep_sound_hrv.sh skripta

Zaključak

U mobilnim aplikacijama OCR tehnologija tek sada dolazi do izražaja. Prijašnji modeli nisu bili u stanju izvoditi sve kompleksne izračune koji se izvode pri OCR obradi. Dosadašnje OCR Iphone/Android aplikacije su posebno prilagođene za mobilne uređaje i većinom koriste neki oblik stabla odlučivanja pri prepoznavanju teksta. Trebalo bi također naznačiti da još ne postoji BADA OCR aplikacija, te bi se aplikacija demonstrirana u ovom radu mogla smatrati prvom takve vrste. Velika mana trenutne OCR aplikacije je činjenica da koristi vanjski server pri OCR/TTS obradi, što je čini ovisnom o HGPRS ili bežičnoj mreži. Sljedeći logičan korak pri razvoju nove verzije bilo bi izvršavanje OCR obrade lokalno na mobitelu. Novi WAVE3 modeli bi bez problema trebali izvoditi navedene funkcionalnosti. Jedna od mogućih izvedbi ove funkcionalnosti jest prevođenje GOCR[53] softvera na mobilnoj platformi. Druga moguća funkcionalnost je posebno prilagođen algoritam koji bi bio u stanju izlučiti tekst s fotografije. Aplikacija demonstrirana u ovom radu koristi funkcionalnost servera pomoću kojeg oslobađa mobilnu platformu od izvođenja preskupih izračuna. Noviji modeli kao Galaxy S2 i iPhone S5 zadovoljavaju hardverske zahtjeve OCR sistema, te se može očekivati zadovoljavajuća primjena lokalno na mobitelu. Do sada je bilo moguće instalirati Tesseract OCR na Android platformi[54], ali izvođenje istog je trajalo predugo vremena.

U ovom radu je opisan OCR na mobilnim tehnologijama. Korišteni okvir koristi već postojeću tehnologiju slobodnog koda. Primjećeno je da fotografije s 640x480 daju najbolje rezultate, te se obrade relativno brzo i efikasno. Prosječno vrijeme prijenosa, korisničke i računalne obrade (ako se koristi bežični pristup internetu) je od 30 do 45 sekundi. Nasuprot tome na Android aplikaciji prosječno vrijeme OCR obrade je oko 3 minute[54]. Predloženi okviri razvoja se mogu primijeniti u velikom broju slučajeva, npr. čitanje poslovnih kartica, turističko pomagalo, prevoditelj itd.

Razvijeni OCR okvir radi u kombinaciji s TTS sistemom otvorenog koda. Aplikacija je projektirana modularno gdje je svaki sustav u potpunosti neovisan jedan od drugog, te je veoma lako promijeniti jednu od korištenih komponenti bez mijenjanja ostalih. Motivacija ovog rada je prikaz općenitog okruženja koje se može koristiti pri izgradnji ambicioznijih aplikacija baziranih na OCR i TTS sustavima.

Literatura

[1] Wikipedia odjeljak koji pojašnjava pojam stvarnosti (2012)

URL: http://en.wikipedia.org/wiki/Reality

[2] Wikipedia odjeljak koji pojašnjava pojam AR – a (2012)

URL: http://en.wikipedia.org/wiki/Augmented_reality

[3] Bonsor Kevin: How Augmented Reality Works (2012)

URL: http://computer.howstuffworks.com/augmented-reality1.htm

[5] Youtube – isječak rada unutar tvorince BMW pomoću AR HUD – a (2007)

URL: http://www.youtube.com/watch?v=P9KPJlA5yds

[6] Tanagram Inc. - službena stranica

URL: http://tanagram.com/

[7] SnapShop Showroom aplikacija – itunes stranica:

URL: http://itunes.apple.com/us/app/snapshopshowroom/id373144101?mt=8

[8] Wikipedia odjeljak o OCR tehnologiji(2012)

URL: http://en.wikipedia.org/wiki/Optical_character_recognition

[9] Wikipedia odjeljak o automatskom prepoznavanju tablica automobila (2012)

URL: http://en.wikipedia.org/wiki/Automatic_number_plate_recognition

[10] Wikipedia odjeljak o sintezi zvuka (2012)

URL: http://en.wikipedia.org/wiki/Speech_synthesis

[11] Službena Stranica ArtoolKIt,

URL: http://en.wikipedia.org/wiki/ARToolKit

[12] Steven Feiner, Blair MacIntyre, Dorée Seligmann: Knowledge-based Augmented Reality for Maintenance Assistance (1995)

URL: http://monet.cs.columbia.edu/projects/karma/karma.html

[13] Neares Wiki itunes stranica

URL: http://itunes.apple.com/us/app/nearestwiki/id331305547?mt=8

[14] SixthSense aplikacija, službena stranica

URL: http://fluid.media.mit.edu/people/pranav/current/sixthsense.html

[15] Layar aplikacija, službena stranica

URL: http://www.layar.com/

[16] Yelp aplikacija, itunes stranica

URL: http://itunes.apple.com/us/app/yelp/id284910350?mt=8

[17] Službena stranica Artoolkit okruženja // Dokumentacija
URL: http://www.hitl.washington.edu/artoolkit/

[19] Prof. dr. Ludvik Gyergyek, dipl. ing, prof. dr Nikola Pavešić, dipl. Ing, prof. dr. Slobodan Ribarić – Uvod u raspoznavanje uzoraka – Izdavačka radna organizacija, Zagreb, 1988

[20] Mohamed Cheriet, Nawwaf Kharma, Cheng-Lin Liu, Ching Y. Suen – CHARACTER RECOGNITION SYSTEMS - A Guide for Students and Practioners - 2007 by John Wiley & Sons

[21] prof. dipl. ing Borko Furht - Handbook of Augmented Reality – Springer Media 2011

[22] T. Kowaliw, N. Kharma, C. Jensen, H. Mognieh, and J. Yao. Using competitive co-evolution to evolve better pattern recognizers. International Journal Computational Intelligence and Applications. 5(3), 305–320, 2005.

[23] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa. Handwritten digit recognition: Benchmarking of state-of-the-art techniques. Pattern Recognition. 36(10), 2271–2285, 2003.

[24] C.-L. Liu, H. Sako, and H. Fujisawa. Handwritten Chinese character recognition: alternatives to nonlinear normalization. In Proceedings of the 7th International Conference on Document Analysis and Recognition, Edinburgh, Scotland, 2003, pp. 524–528.

[25] Tony Karrer: Digital Signal Processor and Text-to-Speech (2010),

http://elearningtech.blogspot.com/2010/06/digital-signal-processor-and-text-to.html

[26] Paul Taylor – Text to Speech Synthesis – University of Cambridge Press, 2009

[27] Miran Pobar, Sandra Martininičić – Ipšić, Ivo Ipšić

Računalni Sustav za Tvorbu Hrvatskog Govora, 2008

[28] Thierry Dutoit: TTS research team, TCTS Lab.

A Short Introduction to Text-to-Speech Synthesis

[29] Steven Zhiying Zhou, Syed Omer Gilani and Stefan Winkler – Open Source OCR Framework Using Mobile Devices – Interactive Multimedia Lab, Department of Electrical and Computer Engineering National University of Singapore

[30] Wikipedia odjeljak o Base64 enkodiranju (2012)

URL: http://en.wikipedia.org/wiki/Base64

[31] Wikipedia odjeljak o POST paketu (2012)

URL: http://en.wikipedia.org/wiki/POST_(HTTP)

[32] Krunoslav Ćosić: Novi val (2012)

URL: http://samsung.mobil.hr/S8530_Wave_2

[33] Wikipedia odjeljak o BADA OS - u (2012)

http://en.wikipedia.org/wiki/Bada

[34] Wikipedia odjeljak IBM NetVista seriji (2012)

http://en.wikipedia.org/wiki/IBM_NetVista

[35] Wikipedija odjeljak o Ubuntu OS – u (2012)

URL: http://en.wikipedia.org/wiki/Ubuntu_(operating_system)

[36] Wikipedija odjeljak o Apache Web serveru (2012)

URL: http://en.wikipedia.org/wiki/Apache_HTTP_Server

[37] Wikipedia odjeljak o Tesseract OCR softveru (2012)

URL: http://en.wikipedia.org/wiki/Tesseract_(software)

[38] Wikipedia odjeljak o MBROLA algoritmu (2012)

URL: http://en.wikipedia.org/wiki/MBROLA

[39] Wikipedia odjeljak o Festival TTS programu (2012)

URL: http://en.wikipedia.org/wiki/Festival_Speech_Synthesis_System

[40] Wikipedia odjeljak o eSpeak TTS programu (2012)

URL: http://en.wikipedia.org/wiki/ESpeak

[41] Službena Bada API dokumentacija, EditArea odjeljak (2012)

URL: http://dpimg.ospos.net/contents/apis/bada/badaV1.0.0a2/

framework/classOsp_1_1Ui_1_1Controls_1_1EditArea.html

[42] Matworks dokumentacija, RGB2GRAY funkcija (2012)

URL: http://www.mathworks.com/help/toolbox/images/ref/rgb2gray.html

[43] Wikipedija odjeljak o Filtru praga (2012)

URL: http://en.wikipedia.org/wiki/Thresholding_(image_processing)

[44] Inadyn DDNS službena stranica

URL: http://www.inatech.eu/inadyn/

[45] Besplatni DDNS server, službena stranica

URL: http://freedns.afraid.org/

[47] Convert Linux naredba, man stranica

http://linux.die.net/man/1/convert

[47] Wikipedija odjeljak o TIFF formatu

URL: http://en.wikipedia.org/wiki/Tagged_Image_File_Format

[48] Tesseract naredba, man stranica

URL: http://www.linuxcertif.com/man/1/tesseract/

[49] textToWave skripta, man stranica

URL: http://dev.man-online.org/man1/text2wave/

[50] Lame naredba, man stranica

URL: http://linux.die.net/man/1/lame

[51] Base64 naredba, man stranica

URL: http://linux.die.net/man/1/base64

[52] Espeak naredba, man stranica

URL: http://www.linuxcertif.com/man/1/espeak/

[53] GOCR, Službena stranica

URL: http://jocr.sourceforge.net/

[54] Tesserac OCR na Android platformi (2012)

URL: http://www.itwizard.ro/interfacing-cc-libraries-via-jni-example-tesseract-163.html

Sažetak

Mobilni telefoni su evoluirali iz jednostavnih jedan prema jedan *(one-to-one)* komunikacijskih uređaja u snažna mala računala koja su u stanju stati u ruku. Danas su novi mobilni uređaji u stanju snimati video, pretraživati Internet i još mnogo toga. Uzbudljive nove aplikacije kao čitač poslovnih kartica, detektori i translatori se pojavljivaju na mobilnom tržištu. Ove aplikacije pomažu ljudima da brzo prikupe informacije u digitalnom formatu, te ih interpretiraju bez da nose laptop sa sobom. Ipak sa svim ovim napredcima još uvijek postoji veoma malo okvira slobodnog koda koji se mogu zadovoljavajuće primijeniti u mobilnoj tehnologiji. Nasuprot tome, postoji velik broj OCR sistema za stolne *(desktop)* platforme, ali nijedan dostupan na mobilnoj. Imajući to na umu, predstavljen je potpuni sistem detekcije i prepoznavanja teksta zajedno sa sintezom govora. U ovom radu razvijen je potpuni OCR/TTS sistem koristeći stolne OCR/TTS programe slobodnog koda, što uključuje OCR sistem zvan Tesseract i eSpeak/Festival TTS.

Ključne riječi: Augmented Reality, OCR, TTS, MBROLA, Mobile devices, Optical Character Recognition, Text-To-Speach, BADA OCR, Open Source

Summary

Mobile phones have evolved from passive one-to-one communication device to powerful handheld computing device. Today most new mobile phones are capable of capturing images, recording video, and browsing internet and do much more. Exciting new social applications are emerging on mobile landscape, like, business card readers, sing detectors and translators. These applications help people quickly gather the information in digital format and interpret them without the need of carrying laptops or tablet PCs. However with all these advancements we find very few open source software available for mobile phones. For instance currently there are many open source OCR engines for desktop platform but none are available on mobile platform. Keeping this in perspective we propose a complete text detection and recognition system with speech synthesis ability, using existing desktop technology. In this work we developed a complete OCR framework with subsystems from open source desktop community. This includes a popular open source OCR engine named Tesseract for text detection & recognition and eSpeak/Festival TTS.

Keywords: Augmented Reality, OCR, TTS, MBROLA, Mobile devices, Optical Character Recognition, Text-To-Speach, BADA OCR, Open Source

Primjeri u kodu

Dolje je naveden izvorni kod mobilne aplikacije. Prikazati ćemo rad funkcija procesuiranja fotografija i komunikacije s serverom. Programski jezik je C++. Pri opisu navedenih funkcionalnosti često će se spominjati *Proccess* funkcija u kojoj se odvija većina logike tijekom obrade fotografija.

CropTool.cpp

Process funkcija kao argument prima pravokutnik. Dimenzije i koordinate navedenog pravokutnika se skaliraju da bi odgovarale fotografiji na kojoj je bio prikazan, nakon čega se preslikavaju svi pikseli iz fotografije na kojoj se nalazio.

```
void CropTool::Process(Rectangle box)
{
    Editor* pEditor = (Editor *) FormManager::GetInstance()-
>GetFormById(Editor::K_SCREEN_ID);

    Bitmap* pDstBmp;
    pDstBmp = new Bitmap();

    Rectangle galleryBound = pEditor->GetImageBound();

    float hScale = float(__pEditBmp->GetWidth()) / float(galleryBound.width);
    float vScale = float(__pEditBmp->GetHeight()) / float(galleryBound.height);
    float scale = Float::Compare(hScale, vScale) > 0 ? hScale : vScale;

    int scaledWidth;
    float w = float(__pEditBmp->GetWidth()) / scale;
    scaledWidth = int(w);
    int scaledHeight;
    float h = float(__pEditBmp->GetHeight()) / scale;
    scaledHeight = int(h);
```

```
	int imgX = (galleryBound.width - scaledWidth) / 2;
	int imgY = (galleryBound.height - scaledHeight) / 2;
	int left = (box.x - imgX) * scale;
	int top = (box.y - imgY) * scale;
	int width = box.width * scale;
	int height = box.height * scale;

	Rectangle cropRect = Rectangle(left, top, width, height);
	result r = pDstBmp->Construct(*__pEditBmp, cropRect);
	if (IsFailed(r))
	{
		AppLog("Bitmap construct failed in CropTool::Process");

		return;
	}

	pEditor->SetCurrentBitmap(pDstBmp);

	pEditor = null;

	Exit();

}
```

RotationTool.cpp

Unutar tijela *Process* funkcije, poziva se *ImageUtil::Rotate* funkcija koja na temelju privatne *__direction* varijable rotira fotografiju.

```
void RotationTool::OnActionPerformed(const Osp::Ui::Control &source, int actionId)
{
	switch (actionId)
	{
```

```
                case K_CW_BTN:
                {
                        SetRotationAngle(IMAGE_ROTATION_90);
                        Process();
                        break;
                }
                case K_ACW_BTN:
                {
                        SetRotationAngle(IMAGE_ROTATION_270);
                        Process();
                        break;
                }
                case K_DONE_BTN:
                {
                        Exit();
                        break;
                }
        }

}

void RotationTool::SetRotationAngle(ImageRotationType dir)
{
        __direction = dir;
}
void RotationTool::Process()
{
        BufferInfo bufInfo;
        ByteBuffer srcBuf, dstBuf;
        Bitmap* pDstBmp = null;
        result r = E_SUCCESS;
        byte* pPixels;
        int bytesPerLine;
```

```
__pEditBmp->Lock(bufInfo);
srcBuf.Construct(bufInfo.height * bufInfo.width * bufInfo.bitsPerPixel / 8);
pPixels = (byte*) bufInfo.pPixels;
bytesPerLine = bufInfo.width * bufInfo.bitsPerPixel / 8;

if (bytesPerLine != bufInfo.pitch)
{
	for (int i = 0; i < bufInfo.height; i++)
	{
		srcBuf.SetArray(pPixels, 0, bytesPerLine);
		pPixels += bufInfo.pitch;
	}
}
else
{
	srcBuf.SetArray(pPixels, 0, bufInfo.pitch * bufInfo.height);
}
__pEditBmp->Unlock();
srcBuf.Flip();
dstBuf.Construct(bufInfo.height * bufInfo.width * bufInfo.bitsPerPixel / 8);

r = ImageUtil::Rotate(srcBuf, dstBuf, Dimension(bufInfo.width, bufInfo.height),
__direction,
		MEDIA_PIXEL_FORMAT_BGRA8888);

Dimension destDim;

TryCatch(r == E_SUCCESS ,, "ImageUtil::Flip failed:%s",
GetErrorMessage(r));

pDstBmp = new Bitmap();

if (__direction == IMAGE_ROTATION_90 || __direction ==
IMAGE_ROTATION_270)
{
```

```
		destDim = Dimension(bufInfo.height, bufInfo.width);
	}
	else
	{
		destDim = Dimension(bufInfo.width, bufInfo.height);
	}
	pDstBmp->Construct(dstBuf, destDim,
BITMAP_PIXEL_FORMAT_ARGB8888, BUFFER_SCALING_NONE);

	CATCH: SetLastResult(r);

	__pEditBmp = Util::CopyBitmapN(*pDstBmp);
	Editor* pEditor = (Editor *) FormManager::GetInstance()-
>GetFormById(Editor::K_SCREEN_ID);
	pEditor->SetCurrentBitmap(pDstBmp);

	pEditor->DrawImageOnCanvas();
	pEditor->RedrawSlider();

	delete pPixels;

	pPixels = null;
	pEditor = null;

}
```

FlipTool.cpp

Process funkcija prima kao argument konstantu *IMAGE_FLIP_HORIZONTAL* i *IMAGE_FLIP_VERTICAL.* Na temelju vrijednosti navedenih konstanti poziva se funkcija *ImageUtil::Flip* koja se brine o promijeni smjera fotografije.

```
void FlipTool::OnActionPerformed(const Osp::Ui::Control &source, int actionId)
{
	switch (actionId)
```

```
	{
		case K_HF_BTN:
		{
			SetFlipDirection(IMAGE_FLIP_HORIZONTAL);
			Process();
			break;
		}
		case K_VF_BTN:
		{
			SetFlipDirection(IMAGE_FLIP_VERTICAL);
			Process();
			break;
		}
		case K_DONE_BTN:
		{
			Exit();
			break;
		}

	}

}

void FlipTool::SetFlipDirection(ImageFlipType dir)
{
	__direction = dir;
}
void FlipTool::Process()
{
	BufferInfo bufInfo;
	ByteBuffer srcBuf, dstBuf;
	Bitmap* pDstBmp = null;
	result r = E_SUCCESS;
	byte* pPixels;
```

```
	int bytesPerLine;

	__pEditBmp->Lock(bufInfo);
	srcBuf.Construct(bufInfo.height * bufInfo.width * bufInfo.bitsPerPixel / 8);
	pPixels = (byte*) bufInfo.pPixels;
	bytesPerLine = bufInfo.width * bufInfo.bitsPerPixel / 8;

	if (bytesPerLine != bufInfo.pitch)
	{
		for (int i = 0; i < bufInfo.height; i++)
		{
			srcBuf.SetArray(pPixels, 0, bytesPerLine);
			pPixels += bufInfo.pitch;
		}
	}
	else
	{
		srcBuf.SetArray(pPixels, 0, bufInfo.pitch * bufInfo.height);
	}
	__pEditBmp->Unlock();
	srcBuf.Flip();
	dstBuf.Construct(bufInfo.height * bufInfo.width * bufInfo.bitsPerPixel / 8);

	r = ImageUtil::Flip(srcBuf, dstBuf, Dimension(bufInfo.width, bufInfo.height),
__direction,
			MEDIA_PIXEL_FORMAT_BGRA8888);
	TryCatch(r == E_SUCCESS, , "ImageUtil::Flip failed:%s",
GetErrorMessage(r));
	pDstBmp = new Bitmap();
	pDstBmp->Construct(dstBuf, Dimension(bufInfo.width, bufInfo.height),
BITMAP_PIXEL_FORMAT_ARGB8888,
			BUFFER_SCALING_NONE);

	CATCH: SetLastResult(r);
```

```
	__pEditBmp = Util::CopyBitmapN(*pDstBmp);

	Editor* pEditor = (Editor *) FormManager::GetInstance()-
>GetFormById(Editor::K_SCREEN_ID);
	pEditor->SetCurrentBitmap(pDstBmp);

	pEditor->DrawImageOnCanvas();
	pEditor->RedrawSlider();

	delete pPixels;

	pPixels = null;
	pEditor = null;

}
```

GrayScaleTool.cpp

Grayscale algoritam se vrti kroz dvije petlje (x i y koordinate), se pristupa pikselima fotografije čije se RGB vrijednosti modificiraju tako da se dobije efekt sive skale.

```
void GrayscaleTool::Process()
{
	if (__pEditBmp == NULL)
	{
		return;
	}

	int width = 0;
	int height = 0;
	int bitsPerPixel = 0;
	int bytesPerPixel = 0;
	BufferInfo info;
```

```
	__pEditBmp->Lock(info);

	width = info.width;
	height = info.height;
	bitsPerPixel = info.bitsPerPixel;

	bytesPerPixel = bitsPerPixel / 8;

	byte gray;
	for (int x = 0; x < width; x++)
	{
		for (int y = 0; y < height; y++)
		{
			int* color = ((int *) ((byte *) info.pPixels + y * info.pitch + x *
bytesPerPixel));
			byte* blue = (byte*) color;
			byte* green = blue + 1;
			byte* red = green + 1;

			gray = ((*blue) * 0.11) + ((*green) * 0.59) + ((*red) * 0.3);
			*red = gray;
			*green = gray;
			*blue = gray;

		}
	}
	__pEditBmp->Unlock();
	Exit();
}
```

ThresholdTool.cpp

Threshold filtar na temelju vrijednosti klizača prolazi kroz sve piksele na fotografiji. Ako je zbroj RGB vrijednosti trenutnog piksela manji od vrijednosti na klizaču onda se njegove RGB vrijednosti postavljaju na 0, 0, 0 odnosno 255, 255, 255 ako je veći.

```
void ThresholdTool::Process() {

	__offset = 0;

	//launch slider
	Editor* pEditor = (Editor *) FormManager::GetInstance()->GetFormById(
			Editor::K_SCREEN_ID);
	pEditor->ActivateSlider(true, this, L"Threshold", 1, 255);

	pEditor = null;

}

void ThresholdTool::OnAdjustmentValueChanged(const Osp::Ui::Control &source,
		int adjustment) {
	ApplyThresholdValue(adjustment);
}

void ThresholdTool::ApplyThresholdValue(int offset) {

	__pProcessBmp = Util::CopyBitmapN(*__pEditBmp);

	if (__pProcessBmp == NULL) {
		return;
	}

	int width = 0;
	int height = 0;
	int bitsPerPixel = 0;
	int bytesPerPixel = 0;
```

```
BufferInfo info;

__pProcessBmp->Lock(info);

width = info.width;
height = info.height;
bitsPerPixel = info.bitsPerPixel;

if (bitsPerPixel == 32) {
        bytesPerPixel = 4;
} else if (bitsPerPixel == 24) {
        bytesPerPixel = 3;
}

double newRed;
double newBlue;
double newGreen;
double scaledoffset = offset * 3;

for (int x = 0; x < width; x++) {
        for (int y = 0; y < height; y++) {
                double* color = ((double *) ((byte *) info.pPixels + y * info.pitch
                                + x * bytesPerPixel));

                byte* blue = (byte*) color;
                byte* green = blue + 1;
                byte* red = green + 1;

                newRed = (double(*red));
                newBlue = (double(*blue));
                newGreen = (double(*green));

                double rgbSUM = newRed + newBlue + newGreen;
```

```
                    if (rgbSUM > scaledoffset) {
                            newRed = 255;
                            newBlue = 255;
                            newGreen = 255;
                    } else {
                            newRed = 0;
                            newBlue = 0;
                            newGreen = 0;
                    }

                    *red = (byte) newRed;
                    *green = (byte) newGreen;
                    *blue = (byte) newBlue;

            }
        }

        __pProcessBmp->Unlock();

        Editor* pEditor = (Editor *) FormManager::GetInstance()->GetFormById(
                    Editor::K_SCREEN_ID);

        pEditor->SetCurrentBitmap(__pProcessBmp);

        pEditor->DrawImageOnCanvas();

        pEditor->RedrawSlider();

        pEditor = null;

        __offset = offset;

}
```

FlipColor.cpp

Flip color algoritam u *Proccess* funkciji prolazi kroz sve piksele, ako je RGB vrijednost trenutnog piksela 255 onda se ona postavlja na 0, inače ako je 0 onda se postavlja na 255.

```
void FlipColorTool::Process() {

	AppLog("--- Flip Color ---");
	__pProcessBmp = Util::CopyBitmapN(*__pEditBmp);

	if (__pProcessBmp == NULL) {
		return;
	}

	int width = 0;
	int height = 0;
	int bitsPerPixel = 0;
	int bytesPerPixel = 0;
	BufferInfo info;

	__pProcessBmp->Lock(info);

	width = info.width;
	height = info.height;
	bitsPerPixel = info.bitsPerPixel;

	if (bitsPerPixel == 32) {
		bytesPerPixel = 4;
	} else if (bitsPerPixel == 24) {
		bytesPerPixel = 3;
	}

	double newRed;
```

```
double newBlue;
double newGreen;

for (int x = 0; x < width; x++) {
        for (int y = 0; y < height; y++) {
                double* color = ((double *) ((byte *) info.pPixels + y * info.pitch
                                + x * bytesPerPixel));

                byte* blue = (byte*) color;
                byte* green = blue + 1;
                byte* red = green + 1;

                newRed = (double(*red));
                newBlue = (double(*blue));
                newGreen = (double(*green));

                int rgbSUM = newRed + newBlue + newGreen;

                if (rgbSUM == 0) {
                        newRed = 255;
                        newBlue = 255;
                        newGreen = 255;
                } else {
                        newRed = 0;
                        newBlue = 0;
                        newGreen = 0;
                }

                *red = (byte) newRed;
                *green = (byte) newGreen;
                *blue = (byte) newBlue;

        }
}
```

```
        __pProcessBmp->Unlock();

        Editor* pEditor = (Editor *) FormManager::GetInstance()->GetFormById(
                        Editor::K_SCREEN_ID);

        pEditor->SetCurrentBitmap(__pProcessBmp);

        pEditor->DrawImageOnCanvas();

        pEditor->RedrawSlider();

        pEditor = null;

        Exit();

}
```

DeleteTool.cpp

```
void DeleteTool::Process(Rectangle box) {

        __pProcessBmp = Util::CopyBitmapN(*__pEditBmp);

        if (__pProcessBmp == NULL) {
                return;
        }

        Editor* pEditor = (Editor *) FormManager::GetInstance()->GetFormById(
                        Editor::K_SCREEN_ID);

        Rectangle galleryBound = pEditor->GetImageBound();
```

```
float hScale = float(__pEditBmp->GetWidth()) / float(galleryBound.width);
float vScale = float(__pEditBmp->GetHeight()) / float(galleryBound.height);
float scale = Float::Compare(hScale, vScale) > 0 ? hScale : vScale;

int scaledWidth;
float w = float(__pEditBmp->GetWidth()) / scale;
scaledWidth = int(w);
int scaledHeight;
float h = float(__pEditBmp->GetHeight()) / scale;
scaledHeight = int(h);

int imgX = (galleryBound.width - scaledWidth) / 2;
int imgY = (galleryBound.height - scaledHeight) / 2;
int left = (box.x - imgX) * scale;
int top = (box.y - imgY) * scale;
int width = box.width * scale;
int height = box.height * scale;

Rectangle cropRect = Rectangle(left, top, width, height);

int x1 = left;
int y1 = top;

int x2 = left + width;
int y2 = top + height;

int bitsPerPixel = 0;
int bytesPerPixel = 0;
BufferInfo info;

__pProcessBmp->Lock(info);

width = info.width;
height = info.height;
```

```
bitsPerPixel = info.bitsPerPixel;

if (bitsPerPixel == 32) {
        bytesPerPixel = 4;
} else if (bitsPerPixel == 24) {
        bytesPerPixel = 3;
}

for (int x = 0; x < width; x++) {
        for (int y = 0; y < height; y++) {
                double* color = ((double *) ((byte *) info.pPixels + y * info.pitch
                                + x * bytesPerPixel));

                byte* blue = (byte*) color;
                byte* green = blue + 1;
                byte* red = green + 1;

                if (x > x1 && x < x2 && y > y1 && y < y2) {

                        *red = (byte) 255;
                        *green = (byte) 255;
                        *blue = (byte) 255;
                }

        }
}

__pProcessBmp->Unlock();

pEditor->SetCurrentBitmap(__pProcessBmp);

pEditor->DrawImageOnCanvas();

pEditor->RedrawSlider();
```

```
	pEditor = null;

	Exit();

}
```

DeleteTool.cpp

Process funkcija kao argument prima pravokutnik. Dimenzije i koordinate navedenog pravokutnika se skaliraju da bi odgovarale fotografiji na kojoj je bio prikazan, nakon čega se sve RGB vrijednosti piksela unutar dimenzija pravokutnika postavljaju na 0.

```
void DeleteTool::Process(Rectangle box) {

	__pProcessBmp = Util::CopyBitmapN(*__pEditBmp);

	if (__pProcessBmp == NULL) {
		return;
	}

	Editor* pEditor = (Editor *) FormManager::GetInstance()->GetFormById(
			Editor::K_SCREEN_ID);

	Rectangle galleryBound = pEditor->GetImageBound();

	float hScale = float(__pEditBmp->GetWidth()) / float(galleryBound.width);
	float vScale = float(__pEditBmp->GetHeight()) / float(galleryBound.height);
	float scale = Float::Compare(hScale, vScale) > 0 ? hScale : vScale;

	int scaledWidth;
	float w = float(__pEditBmp->GetWidth()) / scale;
	scaledWidth = int(w);
```

```
int scaledHeight;
float h = float(__pEditBmp->GetHeight()) / scale;
scaledHeight = int(h);

int imgX = (galleryBound.width - scaledWidth) / 2;
int imgY = (galleryBound.height - scaledHeight) / 2;
int left = (box.x - imgX) * scale;
int top = (box.y - imgY) * scale;
int width = box.width * scale;
int height = box.height * scale;

Rectangle cropRect = Rectangle(left, top, width, height);

int x1 = left;
int y1 = top;

int x2 = left + width;
int y2 = top + height;

int bitsPerPixel = 0;
int bytesPerPixel = 0;
BufferInfo info;

__pProcessBmp->Lock(info);

width = info.width;
height = info.height;
bitsPerPixel = info.bitsPerPixel;

if (bitsPerPixel == 32) {
        bytesPerPixel = 4;
} else if (bitsPerPixel == 24) {
        bytesPerPixel = 3;
}
```

```
        for (int x = 0; x < width; x++) {
                for (int y = 0; y < height; y++) {
                        double* color = ((double *) ((byte *) info.pPixels + y * info.pitch
                                        + x * bytesPerPixel));

                        byte* blue = (byte*) color;
                        byte* green = blue + 1;
                        byte* red = green + 1;

                        if (x > x1 && x < x2 && y > y1 && y < y2) {

                                *red = (byte) 255;
                                *green = (byte) 255;
                                *blue = (byte) 255;
                        }

                }
        }

        __pProcessBmp->Unlock();

        pEditor->SetCurrentBitmap(__pProcessBmp);

        pEditor->DrawImageOnCanvas();

        pEditor->RedrawSlider();

        pEditor = null;

        Exit();

}
```

ImageViewer.cpp

ImageViewer.cpp je daleko najkompleksniji modul u mobilnoj aplikaciji. Njegova uloga je komunikacija s serverom, slanje HTTP paketa, primanje HTTP odgovora, dekodiranja/enkodiranje u Base64 i UTF8. On čini jezgru mobilne aplikacije.

Navedne su najvažnije funkcije koje se koriste pri obradi fotografija:

PrepareImgForUpload – priprema fotografije za slanje na server

GetTextFromServer – primi tekst sa servera

GetSoundFromServer – primi zvuk sa servera

SaveMp3ToMediaSounds – sačuvaj mp3 datoteku lokalno

SaveTxTToMediaOther – sačuvaj tekst datoteku lokalno

EncodeStringToUTF8toBase64 – ekodiraj niz u UTF8

DecodeBase64toUTF8 – dekodiraj Base64 u zvuk

```
result ImageViewer::PrepareImgForUpload(String path) {

	result r = E_SUCCESS;

	Image* pImageDecoder = new Image();

	r = pImageDecoder->Construct();
	if (IsFailed(r)) {
		AppLog("Failed to construct Image");
		return r;
	}

	//String uploadImgPath(L"/Home/upload.jpg");

	Bitmap *pUploadImage = pImageDecoder->DecodeN(path,
			BITMAP_PIXEL_FORMAT_RGB565, Editor::K_IMG_WIDTH,
			Editor::K_IMG_HEIGHT);

	r = pImageDecoder->EncodeToFile(*pUploadImage, IMG_FORMAT_JPG,
path, true);
```

```
	if (IsFailed(r)) {
		AppLog("Failed to encode to upload file");
		return r;
	}

	return r;
}

void ImageViewer::GetTextFromServer(String path) {

	AppLog("--- GetTextFromServer ---");

	String* pProxyAddr = null;
	pMultipartEntity = null;
	pHttpTransaction = null;
	HttpRequest* pHttpRequest = null;

	__pHttpSession = new HttpSession();

	// Create HTTP Session
	__pHttpSession->Construct(NET_HTTP_SESSION_MODE_NORMAL,
pProxyAddr,
			HOST_ADDRESS, null);

	pHttpTransaction = __pHttpSession->OpenTransactionN();

	pHttpTransaction->AddHttpTransactionListener(*this);
	pHttpTransaction->SetHttpProgressListener(*this);

	pHttpRequest = pHttpTransaction->GetRequest();
	pHttpRequest->SetMethod(NET_HTTP_METHOD_POST);
	pHttpRequest->SetUri(TEXT_URI);
```

```
	PrepareImgForUpload(path);

	pMultipartEntity = new HttpMultipartEntity();
	pMultipartEntity->Construct();
	pMultipartEntity->AddFilePart(L"photoFile", path);
	pMultipartEntity->AddStringPart(L"lang", __lang);
	pHttpRequest->SetEntity(*pMultipartEntity);

	pHttpTransaction->SetUserObject(pMultipartEntity);

	pHttpTransaction->Submit();
}

void ImageViewer::GetSoundFromServer(String body) {

	AppLog("--- GetSoundFromServer ---");

	if (body == null || body.GetLength() == 0) {
		AppLog("--- GetSoundFromServer: No text! ---");
		return;
	}

	pMultipartEntity = null;
	pHttpTransaction = null;
	HttpRequest* pHttpRequest = null;

	pHttpTransaction = __pHttpSession->OpenTransactionN();

	pHttpTransaction->AddHttpTransactionListener(*this);
	pHttpTransaction->SetHttpProgressListener(*this);

	pHttpRequest = pHttpTransaction->GetRequest();
	pHttpRequest->SetMethod(NET_HTTP_METHOD_POST);
	pHttpRequest->SetUri(SOUND_URI);
```

```
        AppLog("String to upload: %ls", body.GetPointer());

        pMultipartEntity = null;
        pMultipartEntity = new HttpMultipartEntity();
        pMultipartEntity->Construct();

        String bodyToUpload = EncodeStringToUTF8toBase64(body);

        AppLog("--- bodyToUpload: %ls ---", bodyToUpload.GetPointer());

        pMultipartEntity->AddStringPart(L"body", bodyToUpload);
        pMultipartEntity->AddStringPart(L"lang", __lang);
        pHttpRequest->SetEntity(*pMultipartEntity);

        pHttpTransaction->SetUserObject(pMultipartEntity);

        pHttpTransaction->Submit();

}

result ImageViewer::SaveMp3ToMediaSounds(String filename) {

        result r = E_SUCCESS;

        AppLog("--- SaveMp3ToMediaSounds ---");

        filename.Append(".mp3");

        String sourcePath(L"/Home/");
        sourcePath.Append(filename);

        String destinationPath(L"/Media/Sounds/");
        destinationPath.Append(filename);
```

```
	ContentId contentId;

	ContentManager contentManager;
	contentManager.Construct();

	contentId = contentManager.CreateContent(sourcePath, destinationPath,
			false, null);

	TryCatch(Osp::Base::Uuid::INVALID_UUID != contentId, r =
E_FILE_ALREADY_EXIST, "CreateContent: [E_FILE_ALREADY_EXIST]");

	return r;

	CATCH:

	AppLog("--- CATCH ---");

	AudioContentInfo *audioContentInfo = new AudioContentInfo;
	r = audioContentInfo->Construct(&destinationPath);

	contentId = contentManager.CreateContent(*audioContentInfo);

	r = contentManager.DeleteContent(contentId);

	contentId = contentManager.CreateContent(sourcePath, destinationPath,
			false, null);

	delete audioContentInfo;

	return r;
}

result ImageViewer::SaveTxTToMediaOther(String filename) {
```

```
	result r = E_SUCCESS;

	filename.Append(".txt");

	ContentId contentId;
	ContentManager contentManager;
	DirEnumerator *pDirEnum = null;

	String sourcePath(L"/Home/");
	sourcePath.Append(filename);

	String destinationPath(L"/Media/Others/");
	destinationPath.Append(filename);

	AppLog("--- Other Save dest: %ls ---", destinationPath.GetPointer());

	OtherContentInfo *otherContentInfo = new OtherContentInfo;
	r = otherContentInfo->Construct(&destinationPath);

	contentManager.Construct();

	switch (r) {
	case E_FILE_NOT_FOUND:
		contentId = contentManager.CreateContent(sourcePath,
destinationPath,
				true);
		break;
	case E_FILE_ALREADY_EXIST:
		contentId = contentManager.CreateContent(*otherContentInfo);
		contentManager.UpdateContent(*otherContentInfo);
		break;
	}
```

```
        delete otherContentInfo;
        delete pDirEnum;

        return E_SUCCESS;
}

result ImageViewer::SaveTxTToLocalHome(String filename, String content) {

        filename.Append(".txt");

        File *pFile;
        String filePath(L"/Home/");

        filePath.Append(filename);

        AppLog("--- filepath %ls ---", filePath.GetPointer());

        pFile = new File();
        pFile->Construct(filePath, L"w");
        pFile->Write(content);
        pFile->Flush();
        delete pFile;

        return E_SUCCESS;
}

result ImageViewer::SaveHTTPResponse(void) {

        AppLog("--- SaveHTTPResponse ---");
        EditArea *pEditArea;
        pEditArea = static_cast<EditArea *> (__pPopupResult->GetControl(
                        L"ID_EDIT_HTTP_RESULT"));

        String filename = GetFilenameFromPath(__currentFilePath);
```

```
        SaveTxTToLocalHome(filename, __body);

        SaveTxTToMediaOther(filename);

        String savedMsg = L"Saved in /Media/Others/";
        savedMsg.Append(filename);
        savedMsg.Append(L".txt");

        ShowMsg(savedMsg);

        if (__hasSound) {

                SaveMp3ToMediaSounds(filename);

                savedMsg = L"Saved in /Media/Sounds/";
                savedMsg.Append(filename);
                savedMsg.Append(L".mp3");

                ShowMsg(savedMsg);
        }

        return E_SUCCESS;
}

String ImageViewer::DecodeBase64toUTF8(String body64) {

        String retVal;

        ByteBuffer* pDecodedBuffer = StringUtil::DecodeBase64StringN(body64);

        int limit = pDecodedBuffer->GetLimit();
        pDecodedBuffer->Flip(POSITION_TO_ZERO);
        pDecodedBuffer->SetLimit(limit);
```

```
	Utf8Decoder utf8;
	McharBuffer* pChars = utf8.GetCharsN(*pDecodedBuffer, true);

	StringUtil::MbToString(*pChars, retVal);

	AppLog("--- DecodeBase64toUTF8: %ls ---", retVal.GetPointer());

	delete pChars;

	return retVal;
}

String ImageViewer::EncodeStringToUTF8toBase64(String utf8) {

	AppLog("--- EncodeStringToUTF8toBase64 ---");

	AppLog("--- String to Convert: %ls ---", utf8.GetPointer());

	int byteCount;
	Utf8Encoding utf8Encoding;
	utf8Encoding.GetByteCount(utf8, byteCount);
	ByteBuffer* pBuffer = utf8Encoding.GetBytesN(utf8);

	String encodedStr;
	StringUtil::EncodeToBase64String(*pBuffer, encodedStr);

	AppLog("--- Converted String: %ls ---", encodedStr.GetPointer());

	return encodedStr;
}

String ImageViewer::DecodeBase64toUTF8(String body64) {
```

```
	String retVal;

	ByteBuffer* pDecodedBuffer = StringUtil::DecodeBase64StringN(body64);

	int limit = pDecodedBuffer->GetLimit();
	pDecodedBuffer->Flip(POSITION_TO_ZERO);
	pDecodedBuffer->SetLimit(limit);

	Utf8Decoder utf8;
	McharBuffer* pChars = utf8.GetCharsN(*pDecodedBuffer, true);

	StringUtil::MbToString(*pChars, retVal);

	AppLog("--- DecodeBase64toUTF8: %ls ---", retVal.GetPointer());

	delete pChars;

	return retVal;
}
```