Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Boštjan Krajnc

SPROTNO UVAŽANJE PODATKOV IZ ODJEMALCA SPLETNEGA POKRA

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Peter Peer

Ljubljana, 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00548 / 2013
Datum: 15.9.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **BOŠTJAN KRAJNC**

Naslov: **SPROTNO UVAŽANJE PODATKOV IZ ODJEMALCA SPLETNEGA POKRA**
REAL TIME DATA IMPORT FROM WEB POKER CLIENT

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Naredite načrt in implementacijo programa, ki sproti pridobiva vse potrebne podatke o trenutni poker igri v spletnem odjemalcu. Izhodiščna informacija je slika igralne mize. Locirajte pomembne podatke za sledenje igri ter pretvorite le-te v obliko, ki jo lahko v nadaljevanju uporabi poker robot pri svojem odločanju. Poročajte tudi o učinkovitosti rešitve.

Mentor:
doc. dr. Peter Peer

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Boštjan Krajnc, z vpisno številko **63980243**, sem avtor diplomskega dela z naslovom:

Sprotno uvažanje podatkov iz odjemalca spletnega pokra

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Petra Peera,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 10. marca 2014 Podpis avtorja:

Zahvaljujem se vsem, ki ste, vsak na svoj način, pomagali pri nastanku tega diplomskega dela.

Najprej se zahvaljujem Vam, spoštovani mentor, doc. dr. Peter Peer, za razpoložljivost in usmeritve tekom celotnega procesa izdelave te naloge.

Zahvaljujem se tudi družbi Iskra Sistemi. Hvala Vam, direktor divizije Poslovne rešitve in informatika, g. Miha Praunseis, da ste s tihim razumevanjem sprejemali moje delo na tej nalogi tudi v službenem času.

Posebno zahvalo dajem ženi Maji in staršem, ki ste me v teh trenutkih podpirali in mi stali ob strani.

Vsem, ki jih na tem mestu ne imenujem in ste me v določenem trenutku podprli, pomagali, oziroma mi drugače olajšali delo izrekam: hvala.

Kazalo

Povzetek

V tem diplomskem delu je predstavljen nastanek programa, ki iz poker iger, ki trenutno tečejo na odjemalcu spletnega pokra, sproti pridobiva vse potrebne podatke. Prikazan je postopek lociranja slikovnih podatkov ter pretvorba leteh v podatkovne strukture, ki jih lahko v nadaljevanju uporabi poker robot pri svojem odločanju.

Celoten postopek je sestavljen iz več korakov. Začne se pri iskanju odprtih miz poker odjemalca in zajemu tem mizam pripadajočih slik. V nadaljevanju se z algoritmoma za zaznavanje robov in iskanje obrisov locirajo nekateri podatki. Slike podatkov, ki se pretvarjajo, je potrebno ustrezno predobdelati s pomočjo tehnik kot so binarizacija, prevzorčenje, z uporabo filtrov in morfoloških operacij. Ustrezna predobdelava je izredno pomembna, saj od vseh korakov najbolj vpliva na natančnost optične razpoznave znakov. Za opravljanje optične razpoznave znakov smo izbrali razpoznavalnik Tesseract. Predstavljene so tudi metode odkrivanja in popravljanja napak, ki so nastale tekom optične razpoznave.

Celoten postopek razpoznave podatkov, je bil preizkušen na 149 slikah igralnih miz spletnega pokra. Rezultati natančnosti razpoznave so relativno ugodni, saj je skupna natančnost razpoznave 99,5%. Podane pa so tudi ideje za izboljšanje programa. V smislu izboljšanja uporabnosti programa, pospešitev, kot tudi še bolj kakovostne razpoznave teksta.

Ključne besede: OCR, spletni poker, računalniški vid, zaznavanje robov, iskanje obrisov, predobdelava slik, Open CV, Tesseract

Abstract

The diploma thesis presents development of application, which gathers data from currently opened on-line poker games in real time. It presents procedures for locating picture data and transforming this data into data structures, which can later be used by poker robot to make a decision.

Procedure includes many steps. It starts with finding opened poker tables and capturing corresponding pictures of this tables. In next step edge detection and contour finding operations are applied to find some of the data. Pictures of data to convert needs to be appropriately preprocessed using techniques like binarization, resampling, with use of filters and morphological operations. Good preprocessing is the most important step because it has the biggest impact on optical character recognition accuracy. We used Tesseract OCR engine to do character recognition. Methods to detect and repair errors produced by OCR are also presented.

Program was tested with 149 pictures of on-line poker games. Results are relatively good, overall accuracy of data recognition is 99,5%. Ideas to improve program are also given. Regarding usability, speed and even better quality of text recognition.

Keywords: OCR, on-line poker, computer vision, edge detection, contour finding, picture preprocessing, Open CV, Tesseract

Poglavje 1

Uvod

Poker, že od svojega nastanka, zelo priljubljena igra s kartami, je v začetku 21. stoletja doživel izjemen razcvet. Posnetki velikih tekmovanj se sedaj prenašajo po priznanih televizijskih kanalih, snemajo filme s tematiko pokra, v vsaki trgovini pa že prodajajo pripomočke za igranje pokra. Temu razcvetu je botroval pojav spletnega pokra, ki je igro pripeljal v stanovanja milijonom novih igralcev.

Vsaka tako razširjena in priljubljena igra pa hitro dobi avtomatizirane, računalniške igralce. V svetu pokra takemu igralcu rečemo poker robot ali krajše kar bot. Mnogo svetovno znanih univerz in drugih organizacij med seboj tekmuje, kdo bo izdelal zmagovalnega poker bota. Njihovi izdelki pa se nato spopadajo na različnih tekmovanjih, kot na primer vsakoletnem *The annual computer poker competition* [1]. Tekmovanje poker robotov se je zgodilo tudi na Mariborskem FERI-ju, nazadnje leta 2010.

Obstajajo pa tudi različice poker robotov, ki igrajo proti igralcem preko spleta. Vsak tak program je sestavljen iz treh ključnih delov. Razpoznavalni del je prvi del programa, saj podatke iz poker odjemalca, ki so v obliki animacije na ekranu, pretvori v ustrezne podatke, s katerimi lahko program operira. Drugi del na podlagi dobljenih podatkov in vprogramirane strategije izračuna akcijo, s katero bo igro nadaljeval. Tretji del to akcijo izvede tako, da pritisne gumb na poker odjemalcu in po potrebi vpiše znesek stave. Prvi,

razpoznavalni del programa je predmet in cilj te diplomske naloge.

Pri uresničitvi tega cilja nam bo pomagala optična razpoznava znakov (v angleškem jeziku Optical Character Recognition ali s kratico OCR), veja računalniškega vida. Z uporabo OCR-ja in še nekaterih drugih tehnologij s področja računalniškega vida bomo naredili aplikacijo, ki bo s čim manj napakami preslikala animacijo poker odjemalca v podatkovni model s strukturami, katere lahko kasneje uporabimo za izračun akcije. To so standardne podatkovne strukture tipa znak, tekst, število, vrednost boolean.

Diplomska naloga je razdeljena na šest poglavij. V naslednjem opisujemo osnove igre in pripomočke, ki jih potrebujemo pri igranju spletnega pokra. V tretjem poglavju definiramo probleme, ki jih bomo tekom izdelave reševali in podamo pričakovanja od samega programa. Četrto poglavje poda orodja, s katerimi smo si pri programiranju aplikacije pomagali in opišemo implementacijo aplikacije in rešitve problemov, s katerimi smo se med razvojem srečali. Peto poglavje vsebuje analizo učinkovitosti aplikacije. V šestem poglavju podamo sklepne ugotovitve.

Poglavje 2

Poker

2.1 Zgodovina igre poker

Iz katere igre izvira poker, kot ga poznamo danes, je v veliki meri nejasno. V zgodovini je namreč obstajalo veliko iger, ki so pokru podobne.

Prvi zapisi o pokru podobni igri segajo v začetek 16 stoletja. Igra se je igrala v Španiji, Italiji, Franciji in nekaterih drugih evropskih državah. Vsak igralec je imel v rokah 3 karte, končne kombinacije kart pa so bile podobne današnjim v igri poker, kot so par, tri enake, barva. Nekje do leta 1700 so igri dodali še komponenti stav in blefiranja, iz česar je nastala igra »brag« v Angliji in »pochen« v Nemčiji (kar v prevodu pomeni blefiranje). V Franciji je iz te igre nastali igra »poque«, ki so jo francoski priseljenci prinesli v Ameriški New Orleans.

Prvi, ki je igro pokra opisal v svojih spominih, je angleški potujoči igralec Joseph Crowell. Zapisal je, da so igro igrali leta 1829 v New Orleansu s kupčkom 20 kart. Vsak igralec je v roke dobil 5 kart, nato pa so stavili na to, kdo ima boljšo kombinacijo kart. Takšna igra je popolnoma ista kot »as nas«, perzijska igra iz 16 stoletja, kar še dodatno kaže na to, da so začetki igre v Evropi in Perziji. Vsekakor, pa se je igra poker razcvetela v Združenih državah Amerike.

V 19. stoletju se je igra s pomočjo parnikov na reki Mississippi širila na

sever Amerike, s pojavom zlate mrzlice pa še na zahod. Kasneje je poker dobil veliko različnih variant kot npr. Draw, Stud, Lowball pa tudi Hold'em poker in ostale. [2, 3]

2.2 Zgodovina igre texas hold'em poker

Texas hold'em poker se je iz drugih variant pokra razvil v mestu Robstown v Teksasu, nekje v prvih letih 1900. Ko se je igra razširila po Teksasu, so jo v letu 1960 v Nevado prinesli potujoči kvartopirci kot so Felton McCorkindale, Doyle Brunson, Johnny Moss, Crandell Addington. Dolgo vrsto let je igro v Las Vegasu ponujal le kazino Golden Nugget. Ker so igralci takrat igrali predvsem Five-card stud različico igre se Texas hold'em v ostale, bolj obiskane kazinoje ni razširil. Tako je bilo vse do leta 1969, ko so profesionalni igralci s promocijo te igre začeli v kazinoju Dunes, danes znan pod imenom Bellagio. *Second annual Gambling Freternity Convention* leta 1969 je bil prvi skupek turnirjev, ki je med drugim ponujal tudi igro Texas hold'em poker. Binion's horseshoe je naslednje leto prevzel pravice za ta turnir. Novinar Tom Theckary je organizatorjem predlagal, naj bo igra no-limit Texas hold'em poker celo glavni dogodek na tem vsakoletnem turnirju. Tako je tudi bilo - in je danes še vedno. Udeležba na glavnem dogodku se je skozi vsa leta strmo višala. Od 7-ih prijavljenih igralcev v letu 1970, več kot 100 prijavljenih v letu 1982 pa vse od več kot 6000 prijavljenih v današnjih dneh. Danes je no-limit Texas hold'em poker najbolj igrana, prepoznavna in predvajana različica pokra na svetu. K temu je odločilno pripomogel prihod spletnega pokra leta 1998 in pa prenosi največjih poker turnirjev, kot sta *World series of poker* in *World poker tour* po televiziji. [4, 5]

2.3 Spletni poker

Prvo igro spletnega pokra so odigrali 1. januarja 1998 na odjemalcu Planet poker, ki je bil prvi odjemalec spletnega pokra v zgodovini. Od takrat spletni

poker igra vsakodnevno, ali pa le občasno, več kot 50 milijonov igralcev iz celega sveta. [2] Sama pravila igre so popolnoma ista kot v igri, ki se igra v realnem svetu. Ima pa spletna igra veliko prednosti, seveda pa tudi nekatere slabosti.

Ena od prednosti je na primer ta, da nam ni potrebno iti v kazino, ampak lahko igramo poker kar iz domačega kavča na računalniku, na tabličnem računalniku oziroma celo na novejših mobilnih telefonih. Pogoji za igranje so torej računalnik ali pa katera druga naprava, ki je povezana na splet, aplikacija poker odjemalca in odprt račun pri enem od ponudnikov spletnega pokra. Za igranje s pravim denarjem je seveda potrebno na ta račun naložiti še denar, kar se večinoma uredi s pomočjo spletnih denarnic, ali pa z nakazilom s kreditno kartico. Vsak ponudnik spletnega pokra ponuja tudi brezplačno igranje za točke (angl. fun money), torej učenje.

Igre na spletu se odvijajo velikokrat hitreje, kot v živi igri. Delilec v živi igri ima namreč veliko dela s pobiranjem, mešanjem in deljenjem kart. Na spletu to program naredi v trenutku. V živi igri se v povprečju odigra 30 iger na uro na spletu pa tudi do 100 iger na uro.

Razlika je tudi v tem, da lahko na spletu najdeš igro kadarkoli. Igre namreč tečejo neprekinjeno 24 ur na dan, igralcev pa je pri večjih ponudnikih vedno na pretek, to je več tisoč, deset tisoč igralcev.

Ta količina igralcev in pa uporaba tehnologije omogoča igranje več iger pokra naenkrat, večina poker aplikacij namreč omogoča, da ima igralec odprtih več različnih iger istočasno, kar je v kazinoju nemogoče.

Vsekakor obstajajo še druge prednosti v uporabi spletnega pokra proti igranju pokra v kazinoju a za namene te naloge niso pomembne. Tako kot prednosti pa ima spletni poker tudi nekatere slabosti. Spremljanje nasprotnikovih reakcij je tako možno samo v živi igri. Nekatere akcije v aplikacije niso programirane (recimo odkritje svojih kart, ko izgubiš igro ali pa slepo višanje (angl. live straddle) - prostovoljno višanje igralca, še preden so mu dodeljene lastne karte), ki se v večini primerov uporabljajo v živi igri.

V današnjih časih je najbolj razširjen ponudnik spletnega pokra Poker

stars, na katerem je v vsakem trenutku prijavljenih v povprečju 150.000 igralcev. Ponudnik, katerega odjemalec bom uporabil pri izdelavi te naloge pa je Party poker, ki ima v povprečju 15.000 igralcev za pravi denar.

2.4 Pravila igre no limit texas hold'em poker

No-limit Texas hold'em poker se igra s standardnim kupčkom 52-ih kart. Igralec, ki meša oziroma deli karte ima pred seboj delilčev gumb. Ta gumb označuje najboljšo pozicijo v tej igri, saj igralec, ki ima gumb, opravi akcijo zadnji in pred tem vidi akcije vseh svojih nasprotnikov. Igralec, ki sedi eno mesto levo od delilčevega gumba na mizo položi malo prisilno stavo (angl. small blind), naslednji igralec pa veliko prisilno stavo (angl. big blind). Za ti dve prisilni stavi nato igralci tekmujejo, na mizo pa se položijo še predno se igra začne. Vsak od igralcev dobi dve lastni karti, nato pa igralec levo od gumba začne prvi krog stav. Nato se na mizi odkrijejo 3 skupne karte, ki se jim reče flop. Sledi še en krog stav. Nato se odpre še ena skupna karta, rečemo ji turn. Po še eni rundi stav se obrne še zadnja skupna karta, to je river, kateri sledi še zadnji krog stav.

Možne akcije, ki jih v času stavljenja lahko izvede igralec, ki je na vrsti, so naslednje:

- Odstop (angl. fold) - S to akcijo se igralec odpove nadaljnjemu sodelovanju v trenutni igri in do konca igre ne sme izvesti nobene akcije več.

- Naprej (angl. check) - Akcija se lahko izvede le v primeru, da pred igralca ni bila položena še nobena stava. S tem se stavi odpovemo, akcija pa se prestavi na naslednjega igralca.

- Klicanje (angl. call) - Igralec lahko kliče v primeru, da je bila pred tem na mizo položena stava. S klicem se to stavo izenači, s tem pa igralec pridobi pravico, da nadaljuje z igro.

- Stava (angl. bet) - V primeru, da pred igralcem na mizi ni še nobene stave, lahko stavi. S tem vse preostale igralce prisili, da odstopijo ali pa vsaj kličejo to stavo.

- Višanje (angl. raise) - Kadar je v igri že kakšna stava, je to možno še povišati. Vsi preostali igralci morajo za obstanek v igri plačati povišano stavo.

Višina stav pri no-limit obliki pokra ni omejena. Vsaka stava ali višanje je lahko tudi *all in*, kar pomeni, da igralec stavi vse žetone, ki jih ima pred seboj.

Igralec lahko v igri no-limit Texas hold'em zmaga na dva načina:

- Ali mu s stavo oziroma višanjem uspe vse druge igralce prepričati v odstop.

- Druga možnost za zmago pa je, da se igra odigra do konca, torej da se zaključi zadnji krog stav, zmagovalec pa je tisti, ki ima v rokah najbolj vredno kombinacijo kart. Kombinacija kart mora biti vedno sestavljena iz petih kart. Sestavljena je iz osebnih kart in skupnih kart v poljubni kombinaciji. Lahko se torej uporabi poljubne tri skupne in obe osebni karti, štiri skupne in ena osebna karta. V skrajnem primeru igralec igra mizo (angl. play the board), torej najboljša kombinacija kart ne vsebuje nobene osebne karte in vse skupne karte.

Zmagovalne kombinacije kart si od najmočnejše proti najslabši sledijo v naslednjem zaporedju: kraljeva lestvica, barvna lestvica, poker, *full house*, barva, lestvica, tris, dva para, par, višja karta. Primer je prikazan na sliki 2.1. [6, 7]

Sama pravila igre so sicer preprosta, strategija, ki je na dolgi rok zmagovalna, pa izredno kompleksna.

Slika 2.1: Primeri vseh različnih zmagovalnih kombinacij kart pri pokru

2.5 Aplikacije za pomoč pri spletnem pokru

2.5.1 Odjemalčeva zgodovina odigranih iger

Zgodovina igre (angl. hand history) je zapis poteka celotne igre spletnega pokra. Zgodovino zagotovi odjemalec pokra in je lahko uporabljena pri sporih, ki bi lahko nastali med igro. Vsak poker odjemalec ponuja tudi zapis zgodovine igre, ki se igralcu na lokalni disk zapiše v obliki preproste tekstovne datoteke. Te datoteke igralci uporabljajo za lažje deljenje in nato debatiranje o problematičnih igrah pokra, v večini pa se uporabljajo za vodenje igralčeve statistike odigranih iger. Te datoteke se namreč lahko uvozijo v program kot je Hold'em manager [8] ali pa Poker tracker [9].

Zgodovina igre vsebuje podatke o celotni igri pokra in je z njeno pomočjo mogoče do potankosti rekonstruirati celotno igro. Vsebuje zaporedno številko igre, tip igre in velikost stave, čas odigrane igre, nazive vseh igralcev za mizo in pozicijo, začetno denarno stanje vsakega igralca, akcije, ki so jih igralci izvajali med igro ter karte, ki so bile vidne igralcu, ki je zgodovino zahteval. Zgodovina igre je igralcu dosegljiva šele po koncu vsake igre, tako da sprotnih podatkov iz nje ni mogoče dobiti. Primer zgodovine igre je prikazan na sliki 2.2.

```
Game #12661348108 starts.
#Game No : 12661348108
***** Hand History for Game 12661348108 *****
$25 USD NL Texas Hold'em - Tuesday, February 12, 14:15:17 EST 2013
Table Velocity (Real Money)
Seat 1 is the button
Total number of players : 6/6
Seat 3: And1__x ( $20.52 USD )
Seat 2: ImTheWalrus ( $71.63 USD )
Seat 1: Krabbenmann ( $58.67 USD )
Seat 4: MoneYaaNgel ( $20.85 USD )
Seat 5: clemmesen22 ( $40.72 USD )
Seat 6: kspog ( $58.67 USD )
ImTheWalrus posts small blind [$0.10 USD].
And1__x posts big blind [$0.25 USD].
** Dealing down cards **
Dealt to MoneYaaNgel [ 3h 3c ]
MoneYaaNgel raises [$0.50 USD]
clemmesen22 folds
kspog folds
Krabbenmann folds
ImTheWalrus folds
And1__x calls [$0.25 USD]
** Dealing Flop ** [ 7d, 4h, 4c ]
And1__x bets [$0.50 USD]
MoneYaaNgel raises [$1.91 USD]
And1__x folds
MoneYaaNgel does not show cards.
MoneYaaNgel wins $3.41 USD
```

Slika 2.2: Primer zgodovine poker igre, kot jo zapiše Party poker odjemalec

2.5.2 Hold'em manager

Hold'em manager [8] je za vsakega resnega igralca spletnega pokra nepogrešljiv program. Podobno funkcionalnost ima tudi program Poker tracker [9], izbira med njima pa je stvar igralčevega okusa. Oba programa spadata v kategorijo sledilcev spletnih poker iger. Osnova za delovanja Hold'em managerja so prej omenjene datoteke zgodovine odigrane igre. Program vsako zgodovino odigrane igre, torej ob kreaciji novega zapisa, avtomatično uvozi v svojo podatkovno bazo. S tem igralec pridobi priročen pregled vseh iger, ki jih je kadarkoli odigral na spletu. Ker so v zgodovino igre vključeni popolnoma vsi podatki o določeni igri, je vsako igro za nazaj mogoče pogledati tudi v grafični obliki. Možno je spremljanje napredovanja igralca skozi čas v obliki grafa ali pa tabel. Prav tako omogoča spremljanje različnih statističnih podatkov o igralčevi igri, ki jih program pridobi iz akcij, ki jih je igralec izvedel v preteklih igrah. Med pomembnejšimi so VPIP (Voluntarily Put $ In Pot), ki nam pove v koliko odstotkih igralec kliče oziroma viša stavo pred flopom. V primeru, da je povprečni VPIP 1% pomeni, da igralec stavo pred flopom kliče ali viša samo z najboljšima dvema kombinacijama lastnih kart (AA in KK), kar spada med 1% vseh kombinacij začetnih kart. V nasprotnem primeru, torej VPIP 100% pa da je igralec do sedaj vstopil v vsako igro na flopu z vsemi možnimi začetnimi kombinacijami kart. Med pomembnejšimi podatki so še PFR (Pre Flop Raise), kolikokrat igralec pred flopom v igro vstopi z višanjem, 3Bet, ki pove kolikokrat igralec pred flopom viša že prej povišano stavo, WON SD % prikaže, v koliko odstotkih igralec zmaga na koncu igre in še veliko več. Ti statistični podatki so igralcu spletnega pokra zelo pomembni, saj se tu hitro vidijo napake v strategiji posameznika. Še bolj pa je pomembno, da Hold'em manager preko zgodovine odigranih iger spremlja tudi statistične podatke o nasprotnikih. Le-te lahko na zelo pregleden način izpišemo v bližini vsakega nasprotnika na poker odjemalcu in sicer v obliki skoncentriranega teksta. Temu izpisu rečemo HUD (Heads Up Display).

Slika 2.3: Prikaz izseka poker mize. Desno na sliki je igralec, levo pa temu igralcu pripadajoč HUD.

2.5.3 Hold'em manager HUD

HUD (kratica za Heads Up Display) je del programa Hold'em manager in skrbi za to, da se statistični podatki nasprotnikov prikažejo na poker odjemalcu v njihovi bližini. Prikazani podatki so izračunani iz podatkov, ki jih je program zbral iz vseh dosedanjih iger, ki smo jih odigrali proti temu igralcu. Ker se ob koncu odigrane igre zgodovina avtomatično uvozi v program, se statistika, z vključenimi temi podatki, osveži v začetku naslednje igre.

Bolj izkušeni igralci spletnega pokra istočasno igrajo več iger oziroma miz. Ker je ob tako intenzivni akciji skoraj nemogoče spremljati in si zapomniti, kako vsak nasprotnik igra, je HUD skorajda obvezen pripomoček. V veliko primerih se izkaže, da za pravilno strategijo v igri sploh ni potrebno gledati drugih dejavnikov, kot so npr. karte v roki, pozicija ali pa število žetonov igralcev, ampak samo statistiko s HUDa.

HUD je poljubno nastavljiv. Vsak posameznik si sam izbere in razporedi statistične podatke, ki naj jih prikazuje. HUD razporeditev statističnih podatkov, ki jih uporabljamo je prikazana na sliki 2.3. Podatki v zgornji vrstici od leve proti desni predstavljajo VPIP, PFR, Steal, Fold to steal, Number of hands, spodnja vrstica pa vsebuje podatke CBet, Fold to CBet, 3Bet, Fold to 3Bet, WON SD %.

Glede na to, da so podatki na HUD-u povprečja prejšnjih zgodovin iger in ne trenutne igre, je statistične podatke, ki temeljijo na trenutnih vrednostih

nanj nemogoče vključiti.

2.5.4 Poker robot

Poker robot oziroma skrajšano poker bot je računalniški program. Ta lahko samostojno, brez vmešavanja igralca, igra spletni poker. Velika prednost poker bot-a proti človeškemu igralcu je ta, da robot nikoli ne postane utrujen. Poker lahko v teoriji igra brez prestanka. Človeški igralec zaradi utrujenosti ali pa različnih stresnih situacij začne delati napake, poker bot pa igra s konstantno zanesljivostjo. Ker ne pozna čustev je neranljiv za stresne situacije. [10] Učenje nove strategije pri poker botu pomeni dodatnih par vrstic programske kode, da pa se človeku neka nova strategija prenese v podzavest pa je potrebno dolgotrajno ponavljanje podobne situacije. Popolna predanost programski kodi pa je obenem tudi slabost poker bota proti človeškemu igralcu. Človek z razmišljanjem veliko lažje odkrije nasprotnikov blef kot današnji poker boti. Prav tako se veliko lažje ustrezno prilagodimo strategiji nasprotnika s tem, da spremenimo lastno osnovno strategijo igranja.

Menimo, da so današnji poker boti, na katere smo občasno naleteli v igri, v večini zelo slabi igralci. Igrajo predvidljivo, proti vsem tipom igralcev enako, kar je zelo lahko obrniti proti njim. Do teh napak pa pride zato, ker noben komercialni poker bot ne uporablja statističnih podatkov nasprotnikov, ki jih lahko vidimo na HUD-u. Ti podatki namreč določajo, na kakšen način nasprotnik igra in kje dela napake. Le na ta način in s temi podatki lahko poker bot torej uporabi, vsakemu nasprotniku posebej, primerno strategijo in izkoristi njegove ponavljajoče se napake v strategiji.

Poglavje 3

Definicija problema

Naloga, ki smo si jo zadali, je, da s programom, ki ga bomo razvili, sproti dobivamo vse potrebne podatke o poker igri, ki se trenutno odvija. Z besedo sproti tu označujemo konstantno, v nekem relativno kratkem intervalu. Na primer vsako sekundo, če pa se bo izkazalo, da bo program za obdelavo potreboval več časa pa lahko tudi več sekund.

Program bo deloval v operacijskem sistemu Windows, vhodna informacija s katero bo upravljal, bo ekranska animacija oziroma slika. Bolj natančno, poker odjemalec in pripadajoče mize z igro, ki so prikazane na ekranu.

Izhod programa bodo podatki, ki predstavljajo stanje poker odjemalca v določenem trenutku. Ti podatki morajo biti v obliki primernih podatkovnih struktur za nadaljno uporabo. Torej teksti, znaki, števila, vrednosti boolean.

Podatki z mize, ki jih mora program razpoznati so prikazani na sliki 3.1 v rdečih okvirjih. Ti podatki so:

1. Ime mize in tip igre.

2. Velikost stav na mizi. Velikost velike in male prisilne stave.

3. Zaporedna številka igre.

4. Lokacija delilčevega žetona. Ta je vedno pri enemu od šestih igralcev.

5. Vsi žetoni v potu. Podatek je seštevek podatka žetoni v potu in trenutnih stav vseh igralcev.

6. Žetoni v potu. Žetoni, ki čakajo zmagovalca. V ta podatek še niso vštete stave, ki so prišle na mizo v tem krogu.

7. Skupne karte. Potrebno je detektirati barvo (srce, kara, križ, pik) in velikost (od A do K) vseh petih skupnih kart.

8. Aktivnost igralca v trenutni igri. Če so karte pri igralcu prisotne, pomeni, da je igralec v tej igri še aktiven. Podatek je potrebno dobiti za vse igralce.

9. Žetoni igralca. Podatek je potrebno dobiti za še aktivne igralce.

10. Žetoni igralca v igri. Seštevek žetonov, ki jih je posamezni igralec v trenutnem krogu stavil oziroma plačal. Podatek je potrebno dobiti za vse še aktivne igralce.

11. HUD podatki o igralcu. Podatke je potrebno dobiti za vse še aktivne igralce.

12. Naši osebni karti.

13. Obstoj gumbov za akcijo. S tem, ko so na tem mestu prikazani gumbi z možnimi akcijami, dobimo informacijo, da smo na vrsti za izvedbo akcije.

Program mora biti napisan čim bolj splošno. Kljub temu da bo primarni razvoj aplikacije potekal na odjemalcu Party poker, obstaja namreč še veliko drugih poker odjemalcev. Zato naj program uporablja nastavitveno datoteko oziroma programski razred, ki vsebuje nastavitve za določenega odjemalca. Za uporabo programa z drugim poker odjemalcem, naj bo dovolj le izbira, kateri poker odjemalec se bo uporabljal, program pa bo nato uporabljal pravilne nastavitve.

Slika 3.1: Označeni podatki na poker mizi, ki jih mora program ustrezno pretvoriti.

Program mora delovati nad poljubnim številom istočasno odprtih poker iger. Je pa pogoj, da so vse mize v celoti vidne na ekranski sliki in se med seboj ne smejo prekrivati. Delno prekrivanje bo sicer mogoče, ne smejo pa se prekrivati podatki, ki jih bo program obdeloval.

Večina poker odjemalcev omogoča tudi raztegovanje in krčenje oken z igrami, saj jih na ta način lahko igralec prilagodi svojim potrebam glede na velikost ekrana. Problem pa je v tem, da se pri nekaterih odjemalcih koordinate podatkov, ki jih bo program bral, ne spreminjajo v sorazmerju s spremembo velikosti okna. Pri nekaterih se spreminjajo, pri nekaterih le po korakih, pri drugih po neznanih algoritmih. Zato smo se odločili, da bo ločljivost mize vnaprej določena in zapisana v nastavitvah. V primeru, da bi želel program uporabljati tudi na kateri drugi ločljivosti odjemalca, bi za ta primer naredil drug profil v nastavitvah.

Podatki s HUD-a so edini del mize, ki ni vezan na določenega klienta, saj ga na mizo prilepi zunanji program, Hold'em manager. Ker je HUD-om mogoče spremeniti pozicijo, mora program znati najti HUD v bližnji okolici igralčevega sedeža. Podatki, ki jih prikazuje, tip, velikost in barva pisave pa bodo vsaj v začetni verziji programa določeni.

Izhodni podatki programa bodo osnova za dve aplikaciji, ki jih imam namen naknadno razviti. Prva je poker robot, ki bo za razliko od komercialnih robotov, sprotno uvažal tudi zelo pomembne HUD podatke o igralcih. Druga aplikacija pa je spletna aplikacija, ki bo kopijo igrane igre na odjemalcu, prikazala tudi na spletu. V tem primeru se bodo podatki pisali v podatkovno bazo, na osnovi katere se bo sproti izrisovala spletna stran. Večkrat namreč naletimo na skupine, ki pomagajo nekemu igralcu pri spletni igri ali pa spletnemu poker tekmovanju. Komunikacija po večini poteka prek programov kot je Skype ali pa MSN. Ker pa je podatkov o igri, ki jih za pravilno odločitev potrebujemo, veliko, v večini primerov zmanjka časa, da bi igralec vse potrebno pomagačem napisal ali povedal. S to spletno stranjo bodo podatki sproti vidni vsem prisotnim, časa za debato in odločitev pa bo veliko več.

Poglavje 4

Rešitev problema in implementacija

4.1 Osnovno delovanje programa

Celoten program je napisan v programskem jeziku C# in sestavljen iz dveh funkcionalnosti. Prvi del je namenjen začetnemu vpisu podatkov za novega odjemalca v nastavitve in lažjemu testiranju pravilnosti delovanja. Ob kliku na gumb se zgodi zajem vseh poker iger iz zaslona in kreacija slike za vsako mizo posebej. Te slike so v isti velikosti in obliki, kot tiste na katerih se bo v nadaljevanju sprotno izvajala razpoznava elementov. Primer takšne slike je prikazan na sliki 3.1. Z uporabo kateregakoli grafičnega programa lahko na teh slikah poker miz izvemo koordinate, velikosti elementov kot tudi ostale podatke, ki jih je pred začetkom uporabe potrebno vpisati v nastavitve za novega poker odjemalca. Nekateri primeri podatkov iz nastavitev so prikazani na sliki 4.1

S pomočjo teh podatkov program v nadaljevanju naredi razpoznavo elementov nad dobljenimi slikami in elemente pretvori v enostavne podatkovne tipe.

Drugi del programa je srž teme te diplomske naloge. Ta nad mizami poker odjemalca, ki jih igralec igra, ciklično vrši razpoznavo željenih podatkov.

```
string tableFindString = "Hold'em"; // Tekst, ki se pojavi v naslovu vsake poker mize – za
identifikacijo oken

int hudNumOfLines = 2; // Število vrstic v HUD oknu

Size playerStackSize = new Size(70, 15); // Velikost pravokotnika, ki zajema podatek o
številu žetonov igralca

Point[] playerStackPoints = {new Point(260, 105), new Point(495, 105), new Point(710, 255),
new Point(460, 410), new Point(285, 410), new Point(30, 255)}; // Zgornja leva točka
pravokotnika, ki zajema podatek o številu žetonov igralca – za vseh 6 igralcev

Point[] actionButons = {new Point(437, 510), new Point(567, 510), new Point(698, 510)};
        // Zaporedne točke na vseh treh gumbih za akcijo
...
```

Slika 4.1: Podatki iz nastavitev.

Dobljeni podatki se izpisujejo na izhodu programa.

Glavna zanka sprotnega uvoza in razpoznave podatkov v psevdo kodi izgleda takole:

```
while proces ni ustavljen do
    listaOken = najdi okna, ki vsebujejo poker igro;
    listaSlik = iz listaOken pridobi pripadajoče slike
    iger;
    while listaSlik ni prazna do
        razpoznaj podatke na sliki;
        dopolni/spremeni pripadajoče programske objekte;
        posodobi prikaz podatkov na grafičnem vmesniku;
    end
end
```

4.2 Detekcija odprtih poker iger in pretvorba v sliko

Detekcija odprtih oken v sistemu Windows je dostopna preko Windows API klica `EnumWindows(IntPtr hWnd, IntPtr lParam)`. Funkcija vrne veliko število oken, ki nas v tem primeru ne zanimajo, zato je potrebno dobljen rezultat omejiti glede na vidljivost. Zato z drugim API klicem `IsWindowVisible(IntPtr hWnd)` okna omejimo na trenutno odprta okna v sistemu Windows. Ker je na računalniku lahko odprto še kakšno okno v katerem teče druga aplikacija uporabimo še API klic `GetWindowText(IntPtr hWnd, StringBuilder lpString, int nMaxCount)`, s čimer izločimo vsa okna, ki v svojem naslovu ne vsebujejo teksta, ki smo ga določili v nastavitvah za tega odjemalca.

Če nad dobljenimi okni uporabimo še API klic `GetWindowRect(IntPtr hWnd, ref RECT Rect)`, dobimo za vsako okno še vse štiri kotne točke. S pomočjo teh točk pa lahko iz ekranske slike izrežemo slike vseh poker iger. Te slike so osnova za nadaljnjo obdelavo in razpoznavo podatkov z nje.

4.3 Lociranje HUD podatkov

4.3.1 Programska knjižnica Open CV in Emgu CV

Open CV (Open Source Computer Vision) je odprtokodna knjižnica napisana v programskem jeziku C++. Vsebuje funkcije, ki se uporabljajo v računalniškem vidu in so predvsem namenjene realnočasovnemu procesiranju slik in pa strojnemu učenju. Z razvojem knjižnice je leta 1999 začel Intel Research z namenom pohitriti aplikacije, ki so porabile veliko procesorske moči. Glavni cilj pa je bil razviti knjižnico s čim več funkcijami s področja računalniškega vida, ki bi bile glede delovanja kar najbolj optimizirane in lahke za uporabo. Leta 2012 je Open CV prevzela neprofitna organizacija opencv.org, ki nadaljuje z razvojem. Danes knjižnica obsega že več tisoč algoritmov in funkcij, kar programerjem zelo olajša razvoj aplikacij,

kot je naša. Zadnje različice vsebujejo podporo za večino platform. Poleg Windows, Linux, MacOS tudi Android in iOS. Podpira tudi hitrejše delovanje na večjedrnih procesorjih in celo izvajanje zahtevne programske kode na grafičnih karticah. [11, 12]

Za uporabo Open CV knjižnic v drugih programskih jezikih so napisani mnogi vmesniki in ovoji (angl. wrapper). Ovoj za .Net okolje se imenuje Emgu CV. To pomeni, da lahko z .Net okoljem združljivi programski jeziki prosto dostopajo do vseh funkcionalnosti knjižnice Open CV. Ima pa Emgu CV glede na Open CV izboljšan razred za slike, avtomatično zbiranje smeti (angl. garbage collection), možnost generičnih operacij nad točkami slik itd. [13]

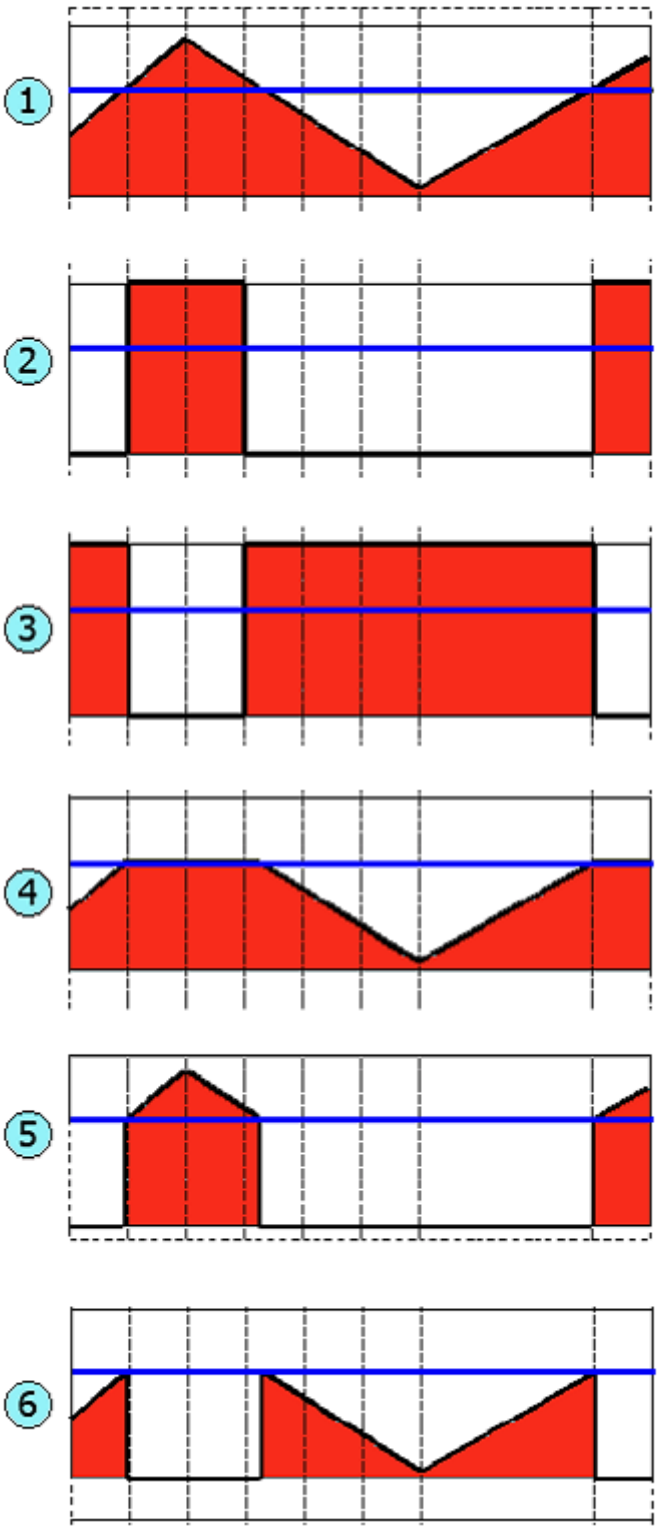
V okviru diplomske naloge bomo knjižnico Open CV uporabili predvsem za predprocesiranje slik pri tekstovni razpoznavi in pa pri iskanju HUD podatkov na sliki.

4.3.2 Binarizacija slike s pragovno segmentacijo

S pojmom binarizacija slike mislimo na pretvorbo originalne slike v sliko, ki vsebuje le dve barvi. Večinoma je to črno-bela kombinacija. Do tega najlažje pridemo z uporabo pragovne segmentacije (angl. threshold segmentation) nad sliko sestavljeno iz različnih vrednosti sivin. S segmentacijo dosežemo izločitev objekta, ki nas zanima, in je v ospredju slike. Do tega pa pridemo s tem, da vsak slikovni element primerjamo s pragom. Tega moramo za uspešno segmentacijo pametno določiti. Ob primerjavi s pragom je vsak slikovni element slike določen za pomembnega in dobi barvo ospredja, ali pa ga označimo kot ozadje. Prag je veliko lažje določiti pri slikah, kjer je ozadje enakomerno obarvano. V našem primeru ozadje nikoli ni komplicirano in ga je lahko ločiti od iskanega predmeta. Pomaga pa tudi to, da je sam HUD popolnoma črne barve in lahko postavimo prag zelo blizu črne barve.

Open CV nam ponuja pet osnovnih vrst pragovne segmentacije, katerih izhod je predstavljen na sliki 4.2, izhod vsakega tipa pa je opisan spodaj [14]:

1. Z rdečo barvo so na y osi predstavljene različne intenzivnosti sivin



Slika 4.2: Grafični prikaz izhoda pragovnih segmentacij Open CV-ja.

$dst(x, y)$. Os x pa predstavlja slikovne elemente sivinske slike. Modra črta predstavlja fiksni prag (angl. threshold). Zgornji $dst(x, y) = MaxVal$ in predstavlja belo barvo, spodnji $dst(x, y) = 0$ pa predstavlja črno barvo.

2. *Threshold binary* – Če je $dst(x, y)$ višja kot prag je vrednost slikovnega elementa $MaxVal$, drugače 0.

3. *Treshold binary inverted* - Če je $dst(x, y)$ višja kot prag je vrednost slikovnega elementa 0, drugače $MaxVal$.

4. *Truncated* – Najvišja $dst(x, y)$ je določena s pragom. Vsem slikovnim elementom, ki imajo višjo intenzivnost se vrednost poreže na vrednost praga.

5. *Threshold to zero* - Če je $dst(x, y)$ nižja od praga je nova vrednost slikovnega elementa 0.

6. *Threshold to zero inverted* - Če je $dst(x, y)$ višja od praga je nova vrednost slikovnega elementa 0.

4.3.3 Canny algoritem za zaznavanje robov

Zaznavanje robov je zbirka matematičnih metod, ki na sliki najdejo slikovne elemente, kjer so spremembe v intenzivnosti svetlosti največje. Ker so te spremembe najbolj vidne na mejah objektov, na ta način pridobimo informacijo o robovih na sliki.

Cannyjev algoritem je ena izmed najboljših metod za zaznavo robov, ki je dobila ime po Johnu Cannyju, njenemu izumitelju. Algoritmi zaznave robov se večinoma uporabljajo v predpripravi slik za nadaljnjo uporabo, predvsem iskanju objektov določenih oblik na sliki.

Canny meni, da mora optimalni algoritem za zaznavo robov zadostiti trem zahtevam:

- Zaznati mora kar največ pravih robov na sliki.

$$\frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Slika 4.3: Gaussov filter za odstranitev šuma.

- Robovi morajo biti označeni kar najbližje robovom na originalni sliki.
- Vsak rob mora biti označen le enkrat, šum pa naj ne povzroča nezaželenih robov.

Algoritem je sestavljen iz več stopenj;

- Glede na to, da je uspešnost zaznave robov odvisna od šuma, je prva stopnja algoritma odstranitev le tega. Za ta namen uporablja Gaussov filter z masko 5×5 elementov (slika 4.3), ki spremeni vrednost slikovnih elementov na sliki. Vsak slikovni element je ponovno definiran kot seštevek vrednosti slikovnih elementov na sosednjih 5×5 lokacijah pomnoženim s pripadajočo Gaussovo utežjo, deljeno s težo celotne maske.
- Naslednji korak je iskanje intenzivnosti in smeri robov. Za ta korak se uporablja Sobelov operator. Za vsak slikovni element na sliki z uporabo

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Slika 4.4: Sobelovi matriki za os x in y.

Sobelovih mask (slika 4.4) izračunamo G_x in G_y. Intenzivnost roba je v neki točki $G = \sqrt{G_x^2 + G_y^2}$. Smer roba pa je $\theta = \arctan \frac{G_y}{G_x}$. Smer roba je zaokrožena na eno izmed vrednosti 0°, 45°, 90°, 135°. [15]

- Naslednji korak je sledenje najdenim robovom glede na prej izračunano moč in smer. Glede na zgornji in spodnji prag, ki sta vhodna parametra v ta algoritem, določi, ali je slikovni element del roba in ga označi z belo barvo, če ni, ga algoritem označi s črno. Če je intenzivnost roba v neki točki večja kot zgornji prag, je ta del roba. Nato pogledamo naslednji slikovni element v izračunani smeri. Če ima ta isto izračunano smer in je intenzivnost roba večja kot spodnji prag, je ta točka prav tako del roba.

- Zadnji korak je brisanje robov z nizko intenzivnostjo, ki so vzporedni robovom z visoko. Ta korak kot izhod poda tanko predstavitev robov. [16]

4.3.4 Iskanje obrisov na sliki

Obris (angl. contour) v tem primeru pomeni skupek točk, ki opisuje neko krivino oziroma lik na sliki. V Open CV-ju je obris predstavljen kot sekvenca, ki vsebuje informacijo o lokaciji naslednje točke na krivini. Funkcija `cvFindContours()` išče obrise nad sliko, ki je nastala kot izhod Cannyjevega algoritma in že vsebuje robove, ki tvorijo iskane krivine in like. Funkcija `cvFindContours()` sprejme tri parametre:

- Način – Možni načini so CV_RETR_EXTERNAL, CV_RETR_LIST, CV_RETR_CCOMP, ali CV_RETR_TREE. Vsak izmed njih pove, katere izmed najdenih obrisov prikazati, na kakšen način in v kakšni strukturi.
- Metoda – S tem izberemo različne algoritme, ki se med seboj razlikujejo v natančnosti dobljenih točk obrisa.
- Tretji parameter je naslov pomnilnika, kamor se obrisi shranijo.

4.3.5 Opis uporabljenih funkcij za detekcijo HUD podatkov

Na sliki 4.5 je prikazana celotna pot do iskane slike HUDa.

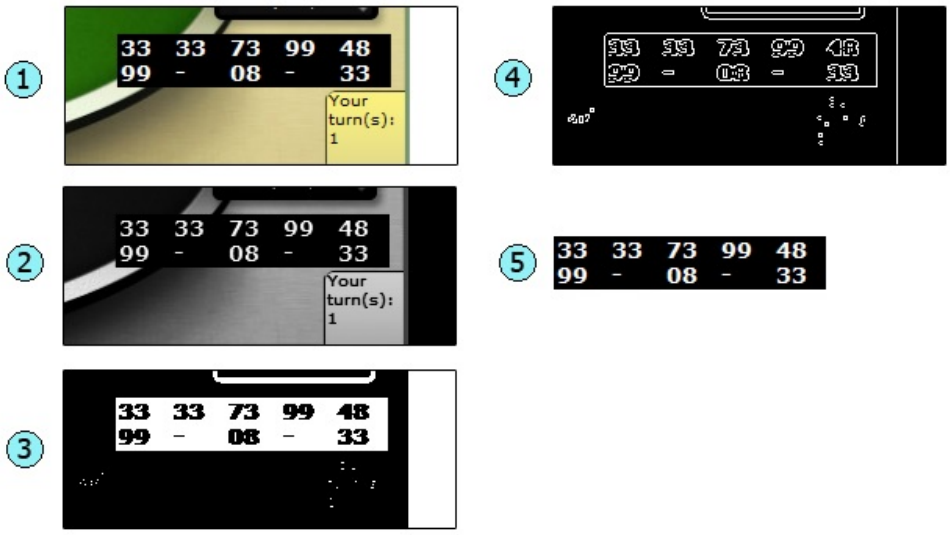
Pod točko 1 je originalna slika izrezana iz celotne slike poker mize. Izrezana je glede na podatke iz nastavitev, kjer je določena velikost in lega področja, kjer se lahko HUD nahaja.

V 2. točki je ta slika spremenjena v sliko sivin (angl. grayscale) in je s tem pripravljena na binarizacijo s pomočjo pragovne segmentacije.

V 3. točki je uporabljena inverzna binarna metoda segmentacije (angl. treshold binary inverted) s parametroma $dst(x, y) = Gray(10)$ in $MaxVal = Gray(255)$.

Točka 4 prikazuje izhod izvedbe Canny algoritma nad prej dobljeno sliko.

Z ukazom `canny.FindContours(Emgu.CV.CvEnum.CHAIN_APPROX_METHOD .CV_CHAIN_APPROX_SIMPLE, Emgu.CV.CvEnum.RETR_TYPE.CV_RETR_TREE, new MemStorage())` dobimo v drevesni strukturi prikazane obrise. Med njimi je tudi obris okvirja HUDa, ki nas zanima. Od ostalih obrisov ga izločimo s tem, da iščemo obris, ki je oblike pravokotnika z določeno širino (v našem primeru je to med 150 in 210 slikovnimi elementi) in višino (med 10 in 50 slikovnimi elementi). Iz najdenega obrisa izvemo koordinate in velikost HUD-a, s tema podatkoma pa ga lahko povsem natančno izrečemo iz originalne slike pod točko 1. Rezultat je slika, ki je prikazana pod točko 5. Ta slika podatkov pa je, kot tudi drugi slikovni podatki, že skoraj pripravljena na optično



Slika 4.5: Postopek iskanja lokacije HUD elementa na sliki po korakih.

razpoznavo znakov. Potrebno jo je še razdeliti na dve polovici z rezom po horizontali. V OCR namreč želimo vstopiti z eno vrstico naenkrat

4.4 Optična razpoznava znakov

Branja in pisanja se večina ljudi nauči v svojih prvih letih vstopa v šolo. Ob koncu otroštva vsak izobražen človek brez problemov bere in piše. Najbolj pomembno je to, da ljudje nismo omejeni na branje in pisanje samo ene vrste pisave. Brati znamo različne tipe tipkanih besedil, ročno napisanih kot tudi pisanih, torej povezane pisave. Beremo lahko tudi znake, ki so delno prekinjeni, okrašeni, pisani krepko ali ležeče ter besedila oziroma besede, ki so pisane od zgoraj navzdol, s črkami različnih barv. Tudi če katera izmed črk oziroma besed manjka, lahko vseeno iz konteksta razberemo, katere so. Kljub več desetletjem razvoja pa imajo računalniki, za razliko od nas, še vedno veliko težav z razpoznavanjem ročne pisave in močno degradiranim tekstom.

Prav tako naredijo veliko napak pri pretvorbi zgoraj opisanih posebnosti. [17]

4.4.1 Zgodovina optične razpoznave znakov

Začetki optične razpoznave znakov (v nadaljevanju označeno z angleško kratico OCR) segajo v začetke 19. stoletja. Iz ameriških patentnih uradov je mogoče razbrati, da so takrat začeli izdelovati prve pripomočke za slepe, ki so temeljili na razpoznavi teksta. Kasneje, leta 1912, je Emanuel Goldberg s svojo napravo pretvoril napisane znake v telegrafske znake. Nekaj let kasneje pa je Fournier D'Albe z napravo Optophone za vsako razpoznano črko zaigral določen zvok. Na ta način so lahko tudi slepi brali oziroma bolje rečeno poslušali napisane tekste. V teh časih je OCR deloval mehansko, s pomočjo optičnih enot in fotocelic.

Leta 1950, s pojavom računalniške ere, pa se je lahko začelo raziskovanje OCR kot ga poznamo danes. Takrat se je začelo izdelovati aplikacije, ki so prebirale strukturirane podatke. David Shepard je leta 1951 razvil enoto, ki jo je poimenoval Gismo. Naprava je znala razpoznati 23 črk ameriške abecede in morsejevo kodo. Gizmo je vplival na celoten nadaljni razvoj OCR aplikacij.

V 60-ih in 70-ih letih prejšnjega stoletja so se masovno začele pojavljati aplikacije, ki so uporabljale OCR. Te aplikacije so uporabljali v bankah, poštah, bolnišnicah, prometu, časopisnih hišah in ostalih industrijskih podjetjih. Predvsem zaradi hitrejšega vnosa podatkov v računalniške sisteme, kot če bi to delo opravljali ljudje. Ameriška pošta je na primer implementirala sistem, ki je avtomatično našel in prebiral naslove na poštnih kuvertah. Celo popraviti je znal napačno črkovane naslove. S tem sistemom pa so omogočil hitro in natančno, predvsem pa poceni usmerjanje pošte k naslovnikom. Vzporedno se je zaradi čedalje hitrejše strojne opreme razvoj OCR sistemov začel tudi v akademskih krogih in raziskovalnih laboratorijih.

Zaradi različnih tipov pisav in slabe kvalitete tiska je pri razpoznavi prihajalo do velikega števila napak. Da bi OCR postal bolj kvaliteten in cenovno

ugodnejši so se proizvajalci zavzemali za standardizacijo na tem področju. Tako sta leta 1968 nastala tipa pisave OCRA in OCRB. Izdelali so jih pri American National Standard Institute (s kratico ANSI) v sodelovanju z European Computer Manufacturers Association (s kratico ECMA). Kmalu zatem jih je v podporo razvoja OCR-ja standardiziral tudi International Standard Organization (s kratico ISO). Posledično je število napak pri razpoznavi zelo upadlo, hitrost pa se je povečala. OCR je postal tudi tako cenovno ugoden, da je popolnoma nadomestil zaposlene, ki so v računalnik pretipkavali podatke.

Nekje v začetku 90-ih se je začelo delati na digitalizaciji knjig in časopisov. Razlog za to početje je računalniško tekstovno iskanje in pa digitalni arhiv takšnega gradiva. V današnjih časih so elektronske knjige (angl. e-book) alternativa tiskanim verzijam ravno zaradi lažje in hitrejše dobavljivosti, lažjega iskanja podatkov v njih in pa tudi zaradi ekološkega vidika. [17, 18, 19, 20]

4.4.2 Trenutno stanje OCR tehnologije

V današnjih časih optična razpoznava latinskega tipkanega teksta ne predstavlja več problema. Natančnost razpoznave teksta je večja od 99%. Vsekakor mora biti za takšno natančnost zagotovljena čista slika, brez šuma, v dovolj veliki ločljivosti, nepoškodovane črke itd. Druga področja, kot so razpoznava ročno napisanih tekstov, tekstov napisanih s pisano pisavo in drugih vrst pisave, kot je na primer azijska pisava pa so še vedno predmet raziskav in izboljšav.

Zadnja leta so, sploh na tabličnih računalnikih, zelo popularni on-line character recognition sistemi. Rečemo jim tudi dinamični CR oziroma pametni CR. Za razliko od OCR sistemov, ki razpoznavajo statične znake, ti spremljajo premike, ki se dogajajo med ročnim pisanjem nekega znaka. Tako bolj kot s samo obliko, znake razpoznavajo z vrstnim redom, hitrostjo in smerjo, s katero so bili posamezni segmenti napisani. Ti sistemi za razpoznavo statičnega teksta niso primerni. Natančnost razpoznave lepo pisane

ročne pisave je nekje v območju od 80% do 90%. Zato je ta tehnologija uporabljena le v zelo omejenem krogu aplikacij, kjer je potrebna hitra razpoznava, napake pa niso zelo pomembne.

Najbolj kritična v teh časih je razpoznava tekstov pisanih s povezano, pisano pisavo (angl. cursive handwriting). Za boljšo natančnost je pri razpoznavi takšnih tekstov potrebno uporabiti kontekstualne in slovnične informacije. [20]

4.4.3 Koraki optične razpoznave znakov

Vsako optično razpoznavo znakov vršimo nad sliko, na kateri so narisani znaki. Takšna slika je vhod v verigo korakov optične razpoznave. Izhod pa je besedilo oziroma znaki, ki so predstavljeni kot računalniški niz znakov.

Proces razpoznave se začne s predobdelavo slike. Željeni izhod iz predobdelave predstavlja črno-bela slika ali pa sivinska slika, ki vsebuje samo tekst, ki ga hočemo razbrati. Da ta izhod dosežemo, imamo na voljo veliko že znanih tehnik, ki so opisane spodaj. Katere od tehnik uporabimo pa je odvisno od kompleksnosti in kvalitete vhodne slike. [21]

Naslednji korak je detekcija teksta oziroma znakov na tej sliki in sama optična razpoznava teksta nad očiščeno sliko.

Na koncu nad dobljenim tekstom naredimo še poobdelavo. Ta korak vsebuje preverjanje dobljenega rezultata glede na pričakovan rezultat. Če na primer vnaprej vemo, da je na nekem mestu možen samo številčni podatek, razpoznani podatek pa je tekst, nas algoritem poobdelave opozori na napako, če pa je možno, lahko napačno razpoznan tekst tudi avtomatično popravi. V primeru razpoznanih besed se lahko te preverjajo in popravljajo z uporabo slovarja. Kadar govorimo o stavkih, ali drugih pomenskih sklopih besed pa se lahko v primeru napačno razpoznane besede, beseda popravi glede na kontekst v besedilu in pa glede na predpostavljeno pravilno razpoznane črke v besedi.

4.4.3.1 Predobdelava slike

4.4.3.1.1 Binarizacija

Proces binarizacije slike s pomočjo pragovne segmentacije je že opisan v poglavju 4.3.1. Opisana je segmentacija z globalnim pragom, saj izbrani prag velja na področju celotne slike. Uporablja se na slikah, kjer je ozadje preprosto, torej enobarvno, ali pa tudi večbarvno, a je intenziteta slikovnih elementov v ozadju dovolj oddaljena od intenzitete slikovnih elementov iskanega objekta. Največkrat uporabljamo Otsu-jevo tehniko. Ta je namreč znana kot najbolj uporabna in najhitrejša. Kadar pa ima objekt v ospredju bolj kompleksno ozadje, uporabimo segmentacijo z lokalnim pragom. Tu se prag izračuna za vsako območje slike, segmentacija s tem pragom pa se nato izvede samo nad slikovnimi elementi iz tega območja. Obstajajo tudi drugi načini segmentacije, ki pa za razločevanje ne uporabljajo direktno praga, ampak delujejo na najdenih robovih. Med najboljše spadajo Parkerjeva metoda, White & Rohrerjeva metoda, Trier & Taxtova metoda. [22]

4.4.3.1.2 Prevzorčenje

Digitalne slike so sestavljene iz matrike slikovnih elementov, ki si jih lahko predstavljamo kot vzorce, ki imajo določeno vrednost. Prevzorčenje pa je tehnika, ki originalno sliko pretvori v novo verzijo slike z različno višino in/ali širino slike. Pri povečanju velikosti slike gre za vzorčenje navzgor, pri zmanjšanju za vzorčenje navzdol. Pri vzorčenju navzgor se število slikovnih elementov sicer poveča, a nova verzija slike ne vsebuje nobene nove informacije, ki jo starejša verzija ne bi imela. Rezultat tega je zamegljena slika. Logična posledica pri vzorčenju navzdol je izguba informacij iz originalne slike. Najbolj znane metode prevzorčenja so naslednje:

- Metoda najbližjega soseda - Vsak slikovni element nove slike ima vrednost najbližjega soseda na originalni sliki.
- Bilinearno - Ta metoda nove slikovne elemente izračuna s pomočjo linearne interpolacije štirih najbližjih točk na originalni sliki.

- Bikubično - Najbližje 4×4 točke iz originalne slike se uporabijo za izračun nove vrednosti. Ta metoda daje najboljše rezultate. [23]

V programu smo tehniko prevzorčenja uporabili praviloma za povečanje velikosti slike. Originalne slike podatkov za razpoznavo so namreč veliko premajhne.

4.4.3.1.3 Procesiranje s histogramom

Histogram nam predstavi pogostost pojavljanja sivin na določeni sliki. To prikaže na razponu od 0, kar predstavlja črno, pa do vrednosti 255, ki predstavlja belo barvo. Procesiranje slik s pomočjo histograma nam lahko izredno izboljša videz slike, predvsem pa nam olajša binarizacijo s pomočjo praga. Histogram nam ne da informacije o prostorski razporeditvi slikovnih elementov na sliki, lahko pa iz njega razberemo:

- Kontrast: Opredeljuje kombinacijo vrednosti razpona intenzitetnih vrednosti, ki so uporabljene na sliki in razlike med najvišjo in najnižjo vrednostjo.

- Dinamično območje: Razlika med najvišjo in najnižjo tonsko vrednostjo.

Z metodo izenačevanje histograma (angl. histogram equalization) dosežemo bolj enakomerno razporejenost intenzitet na histogramu. Ta metoda območjem s slabim kontrastom na sliki omogoči večji kontrast.

Metoda raztezanja histograma (angl. histogram stretching) pa nad celotno sliko poveča kontrast tako, da najtemnejši slikovni element izvorne slike dobi najnižjo vrednost na histogramu, najsvetlejši pa najvišjo vrednost. Ostali slikovni elementi se vmes linearno razporedijo. [21]

V samem programu uporaba histograma sicer ni bila potrebna. Kontrast med tekstom in ozadjem je pri odjemalcu Party poker pri vseh tipih podatkov dovolj visok. Je pa uporaba histograma nepogrešljiva pri kakovostni binarizaciji slike z nizkim kontrastom.

$1/9 \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ $1/16 \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$

Slika 4.6: Dva primera matrike, ki se uporabljata pri povprečnem filtriranju.

4.4.3.1.4 Filtriranje

Povprečni (angl. mean) filtri zmanjšajo intenziteto sosednjim točkam. Uporabljajo se za glajenje slik. Takšnim filtrom rečemo nizkoprepustni filtri (angl. low pass). Vsakemu slikovnemu elementu določi novo vrednost, ki je povprečna vrednost sosednjih slikovnih elementov. Uporabljajo se matrike velikosti 3×3, 5×5, ... Večja kot je matrika, bolj intenzivno glajenje dobimo. Primera matrik sta na sliki 4.6.

Gaussov filter je namenjen glajenju slik. Večji poudarek daje slikovnim elementom, ki niso na robu. Zato je v večini uporabljen na sliki, preden na njej izvajamo funkcijo iskanja robov. Na ta način se namreč znebimo odvečnega šuma, ki bi kasneje povzročal nezaželene robove. Primer maske je prikazan na sliki 4.3.

Visokoprepustni filtri (angl. high pass) so namenjeni ostrenju slike. Delujejo na isti način kot nizkoprepustni filtri, le da uporabljajo drugačno matriko. Vsakemu slikovnemu elementu se določi nova vrednost, ki je vsota vrednosti sosednjih slikovnih elementov, pomnoženih z utežjo v matriki. Če ima določen slikovni element enako intenziteto kot njegovi sosedje iz pripadajoče matrike, se vrednost ne spremeni, če pa je intenzivnost drugačna, se intenzivnost ojača. Dva primera matrik sta prikazana na sliki 4.7.

Filter mediane je namenjen odstranjevanju binarnega šuma (angl. salt and pepper). Je poseben, nelinearni tip nizkoprepustnega fitra. Uporabi se matrika, ki se uporabi kot okno za računanje nove vrednosti slikovnega elementa. Nova vrednost je sredinska vrednost po sortiranju vseh sosednjih

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

Slika 4.7: Primera matrike za visokoprepustno filtriranje.

123	127	150	120	100
119	115	134	121	120
111	120	122	125	180
111	119	145	100	200
110	120	120	130	150

		121		

Slika 4.8: Primer matrike filtra mediane. Izmed sortiranih vrednosti (100, 115, 119, 120, 121, 122, 125, 134, 145) je sredinska vrednost 121.

slikovnih elementov pripadajoče matrike. Primer je prikazan na sliki 4.8. [21]

4.4.3.1.5 Morfološke operacije

Segmentacija lahko na sliki povzroči tudi nekatere efekte, ki si jih ne želimo. Nekatere slikovne elemente lahko interpretira kot ozadje in jih zato odstrani iz slike. Do tega ponavadi pride, kadar je prag pri segmentaciji izbran previsoko. Takšne večje luknje lahko znak razbijejo na več delov. Nasprotno, kadar je prag izbran prenizko, lahko pride do združevanja znakov, kar kasneje pri razpoznavi povzroča težave. Takšne probleme rešujemo z uporabo morfoloških operacij. Uporabne operacije so erozija, dilatacija, odpiranje, zapiranje in zoževanje: vse te tehnike se uporabljajo na binarnih slikah:

- Erozija in dilatacija - Erozija zmanjša obseg objekta tako, da odstrani

0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	255	255	255	255	0	0		0	0	0	255	255	0	0	0
0	0	255	255	255	255	0	0		0	0	255	255	255	255	0	0
0	0	255	255	255	255	0	0		0	0	255	255	255	255	0	0
0	0	255	255	255	255	0	0		0	0	0	255	255	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

Slika 4.9: Primer erozije s pragom 3.

0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	255	255	0	0	0
0	0	255	255	255	255	0	0		0	0	255	255	255	255	0	0
0	0	255	255	255	255	0	0		0	255	255	255	255	255	255	0
0	0	255	255	255	255	0	0		0	255	255	255	255	255	255	0
0	0	255	255	255	255	0	0		0	0	255	255	255	255	0	0
0	0	0	0	0	0	0	0		0	0	0	255	255	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

Slika 4.10: Primer dilatacije s pragom 2.

slikovne elemente na robovih objekta, dilatacija pa obratno, poveča obseg objekta tako, da na robovih doda dodatne slikovne elemente. Obe operaciji za delovanje potrebujeta prag. V primeru erozije na vsakem belem (z vrednostjo 255) slikovnem elementu slike algoritem preveri vse sosednje elemente. Algoritem preveri, če je črnih (z vrednostjo 0) sosednjih slikovnih elementov več kot je določen prag. V tem primeru slikovni element spremeni v črno. Pri eroziji je proces obraten. Algoritem za vsak slikovni element, ki ima vrednost 0, preveri število sosedov z vrednostjo 255. Če je teh več kot določa prag, se slikovnemu elementu spremeni vrednost v 255. (sliki 4.9 in 4.10)

- Odpiranje in zapiranje - Operaciji sta podobni eroziji in dilataciji, le da objektov, nad katerima delujeta, ne spremenita v tako velikem obsegu. Odpiranje razveže objekte, ki so med seboj rahlo povezani. Prav tako

poveča obseg lukenj znotraj objektov. Zapiranje poveže objekte, ki so med seboj minimalno oddaljeni in zapolni nezaželene luknje v objektih.

- Zoževanje - Je proces, ki obliko objekta spremeni v obliko njegovega okostja. S tem se sicer ohrani dolžina in širina objekta, njegova originalna povezljivost. Operacijo si lahko predstavljamo kot večkratno izvajanje erozije nad objektom, dokler ta nima debeline okostja - samo en slikovni element. Pri tem pa nikoli ne sme priti do prekinitve v objektu. [21]

Razen podatka z imenom mize, so vsi podatki Party poker odjemalca zapisani v krepki obliki. Problem elegantno rešimo z izvedbo operacije erozije nad sliko s krepkim tekstom. Praktičen primer je prikazan na 4.11 v zadnji vrstici.

4.4.3.1.6 Uporaba predprocesiranja slik

Predprocesiranje je potrebno izvesti na vsaki sliki, na kateri se kasneje vrši razpoznava teksta. Uporaba predprocesirnih tehnik v programu je prikazana na sliki 4.11. Razpoznavo, torej pretvorbo teksta narisanega na sliki v tekst, bo v našem primeru opravljal razpoznavalnik Tesseract, ki je bolj podrobno opisan v naslednjem poglavju. V primeru, da bi v razpoznavalnik vnašali originalne slike, izrezane iz slike mize, bi bila natančnost razpoznave le nekaj odstotna. Da lahko pravilno obdelamo sliko, je seveda potrebno vedeti kakšno sliko razpoznavalnik sploh pričakuje.

Pomembna sta tako ločljivost kot tudi velikost teksta. Natančnost razpoznave začne padati pri velikosti teksta 10 slikovnih točk. Kadar ima tekst višino pod 10 slikovnih elementov je za pravilno razpoznavo zelo malo možnosti, pod 8 slikovnih elementov pa je večina teksta izločena kot šum.

Najboljša razpoznava se izvede na bitnih slikah z belim tekstom na črni podlagi. Tesseract deluje sicer tudi nad sivinskimi in celo barvnimi slikami, a natančnost s tem pada.

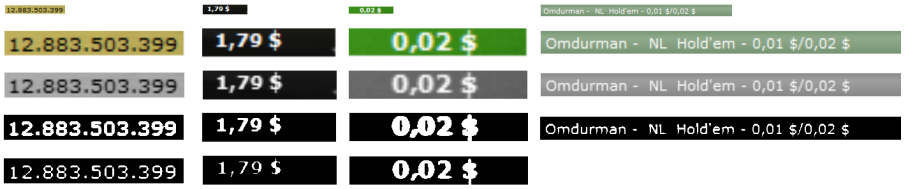
Normalno zapisan tekst zagotavlja najboljše rezultate. Če je pisava poševna, trdo pisana ali podčrtana, je potrebno računati na padec natančnosti. [24]

Na sliki 4.11, v prvi vrstici, so primeri štirih originalnih slik podatkov, ki jih je potrebno razbrati. Na sliki 4.12 je eden izmed teh povečan, da se vidijo tudi posamezni slikovni elementi. Glede na to, da je višina teksta le 7 slikovnih elementov, je razpoznava nad takšno sliko nesmotrna. Zato se vsaka slika najprej prevzorči na večjo velikost. Najprej se izračuna koeficient $koef = \frac{50\ slikovnih\ elementov}{visina\ originalne\ slike\ v\ slikovnih\ elementih}$. Velikost nove slike je s tem koeficientom pomnožena originalna višina in širina. Višina okrog 50 slikovnih elementov se je izkazala kot dovolj velika. Če bi generirali večjo sliko bi bila to potrata računalniških virov, procesiranje vseh naslednjih korakov pa na večjih slikah traja občutno dlje. V drugi vrstici slike 4.11 je viden rezultat omenjenega prevzorčenja.

Druga operacija, ki se izvede nad vsemi slikami, je sprememba v sivinsko sliko, iz te pa z uporabo pragovne segmentacije v binarno, črno-belo sliko. Prag z vrednostjo 135 se je obnesel na vseh slikah. Program sicer ponuja v nastavitvah za vsak tip podatka vnesti svoj prag, če pa ta ni vpisan, se vzame privzeti, torej z vrednostjo 135. Uporabljena vrsta segmentacije je na vseh slikah *threshold binary*, ki je prikazan pod točko 2 na sliki 4.2. Izjema so slike s podatkom o velikosti kart in zaporedni številki igre. V teh primerih se namreč pojavlja temnejši tekst na svetli podlagi, zato je v teh primerih uporabljen *threshold binary inverted* tip segmentacije (točka 3 na sliki 4.11). Tesseract sicer razbere tudi črn tekst na beli podlagi, a smo se tu vseeno držali smernic avtorjev. Primeri sivinske in binarne slike nekaterih podatkov so v tretji in četrti vrstici na sliki 4.11.

Pravilnost razpoznave je večja, če na večini slik uporabimo še funkcijo erozije. Ta znake stanjša in jih naredi Tesseractu lažje berljive, natančnost OCR-ja pa se poveča. Izjema je ime mize (zadnja kolona na sliki 4.11), ki za razliko od ostalih podatkov že v originalu ne uporablja trde pisave. Zato se nad tem podatkom funkcija erozije ne izvaja. Erozija je na primerih prikazana na sliki 4.11 v zadnji vrstici.

Na odjemalcu Party poker drugih tehnik predprocesiranja nismo uporabili. Izhodne slike namreč ustrezajo vsem zahtevam Tesseracta. Znaki med



Slika 4.11: Postopki predobdelave slike za nekatere podatke z mize.

Slika 4.12: Originalna slika teksta za razpoznavo povečana na nivo slikovnih elementov.

seboj niso povezani, deformacij znakov ni, tekst pa je lepo berljiv. Tudi ostale tehnike predprocesiranja bomo najbrž uporabili pri razpoznavi podatkov za katerega drugega odjemalca. Le-teh je namreč še veliko, prav vsak od njih pa ima drugačno grafično predstavitev podatkovnih elementov.

4.4.3.2 Optična razpoznava znakov

4.4.3.2.1 Zgodovina razpoznavalnika Tesseract

Tesseract je OCR razpoznavalnik, ki ga je leta 1984 začel rezvijati Hewlett Packard. Razvijati se je začel kot dodatek za Hp-jeve skenerje, saj je v tistih časih večina komercialnih OCR pogonov znala razbrati le najlepše natisnjen tekst. Leta 1995 je bil med prvimi tremi pogoni na tekmovanju *UNLV Annual Test of OCR Accuracy*. V nadaljnih letih, vse do 2005 se Tesseract ni veliko spreminjal, zato so njegovo kodo sprostili kot odprto kodo. Od takrat naprej razvoj vodi in financira Google in je prosto dosegljiv na `http://code.google.com/p/tesseract-ocr`. Še vedno je eden najboljših odprtokodnih razpoznavalnikov v tem trenutku. [25]

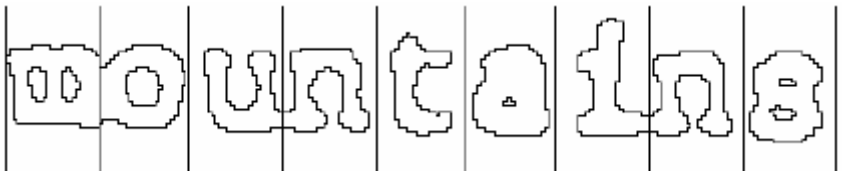
4.4.3.2.2 Arhitektura Tesseracta

Hewlett Packard je ločeno od razvoja Tesseracta razvijal tudi povezan sistem, ki je analiziral postavitve elementov na sliki. Ker to ni del odprte kode Tesseracta, ta kot vhod pričakuje sliko, ki vsebuje samo tekst, ali pa sliko, ki vsebuje tudi druge elemente, a ima definirane regije, kjer se nahaja tekst za razpoznavo. Celoten proces se izvaja po vnaprej določenem zaporednem izvajanju določenih funkcij. Prva funkcija je iskanje in analiza povezanih komponent, kjer se opisi linij shranijo v tekstovne bloke (angl. blobs). Tekstovni bloki se nato povežejo v vrstice teksta. Vrstice se glede na presledke razdelijo v besede, če pa je tekst pisan z enako širokimi znaki, se besede razdelijo še na znake. Razpoznava od tu naprej poteka v dveh ciklih. V prvem ciklu se poizkuša razpoznati posamezne besede. Vsaka zadovoljivo razbrana beseda se vpiše v prilagodljiv klasifikator kot že naučena oziroma relevantna beseda. Če kasneje proces naleti na isto besedo, s tem potrdi bolj natančno razpoznavo te besede. Ker se besede na dnu strani v prilagodljivi klasifikator vpiše prepozno, da bi pomagale pri razpoznavi besed višje v tekstu, se v drugem ciklu proces ponovi še enkrat od začetka do konca strani. Nazadnje se obravnavajo še mehki (angl. fuzzy) presledki in majhne velike tiskane črke (angl. small-caps). [25]

4.4.3.2.3 Moduli Tesseracta

Tesseract je sestavljen iz treh modulov:

- Iskanje vrstic , besed in znakov - Algoritem za iskanje vrstic je zasnovan tako, da ukrivljenih vrstic ni potrebno ravnati. Zaradi tega ne izgubimo kvalitete originalne slike. Iskanje vrstic Tesseract izvede glede na prazno horizontalno vrzel med vrsticami. Ob tem izpusti manjšinske znake, ki to vrzel motijo, kot so ločila, apostrofi in šum. Vsaki vrstici se določi osnovna linija, na katero se razvrstijo tekstovni bloki, ki pripadajo določeni vrstici. Tesseract nato v vsaki vrstici preveri, če imajo znaki v vrstici fiksno širino. Če najde takšen tekst, ga razreže z uporabo znane širine. Takšen primer je prikazan na sliki 4.13. Na



Slika 4.13: Razrez po znakih s fiksno širino.

ta način dobimo celoten tekst razrezan v vrstice, besede in znake. Če tekst ni zapisan v fiksni širini, se za razrez uporabljajo bolj kompleksni algoritmi, ki iščejo horizontalne praznine med besedami.

- Statični klasifikator - Neznani znak, ki ga iščemo, se najprej razdeli v več manjših smiselnih delov imenovanih bitni vektorji, ki jih predstavlja x in y koordinata ter podatek o kotu. Na podlagi teh se sestavi kratka lista kandidatov, ki predstavljajo možne vrednosti neznanega znaka. Nato se neznani znak po podobnosti primerja s prototipi iz te liste. Najbolj podoben znak je nato izbran kot izhod iz statičnega klasifikatorja.

- Prilagodljivi klasifikator - Statični klasifikator se mora obnesti dobro v splošnem, torej na vsakem tipu pisave. Medtem pa je prilagodljivi klasifikator bolj prilagojen istemu tipu pisave. Prednost tega je očitna, saj se v posameznih dokumentih ponavadi uporablja le malo vrst pisav, v večini samo ena. Oba tipa klasifikatorjev se med seboj sicer razlikujeta samo po tipu normalizacije znakov in bazi naučenih znakov. Statični klasifikator normalizira pozicijo znakov glede na centroid ter velikost. Prilagodljivi klasifikator pa normalizira znake na osnovi osnovne linije in višine, kot je prikazano na sliki 4.14. [25]

4.4.3.2.4 Uporaba Tesseracta

Uporaba Tesseracta nad ustrezno predobdelano sliko je preprosta. Za inicializacijo objekta Tesseract je potrebno izvesti naslednjo vrstico:

Slika 4.14: Normalizacija znakov statičnega klasifikatorja na levi in dinamičnega na desni.

```
Tesseract Ocr = newTesseract("tessdata",
"eng",Tesseract.OcrEngineMode.OEM_DEFAULT).
```

Parametri, ki vstopajo v konstruktor, so pot do Tesseracta na lokalnem disku, jezik, ki se bo pri razpoznavi znakov uporabljal in način delovanja Tesseracta. Tesseract namreč lahko razpoznava tudi znake jezikov kot sta kitajščina in arabščina. Možno pa ga je tudi naučiti novega jezika. S preprostima spodnjima ukazoma nato v spremenljivko izhod zapišemo razpoznani tekst.

```
Ocr.Recognize(slika_s_tekstom.png);
string izhod = Ocr.GetText();
```

Tesseract sam odloča, katere module izvaja med razpoznavo teksta. Na delovanje lahko sicer vplivamo preko nastavitev v njegovi nastavitveni datoteki. Te se prenastavljajo le v izjemnih primerih, v našem primeru so vse nastavitve privzete, saj so teksti za razpoznavo relativno preprosti. Tudi v praksi se je izkazalo, da daje privzet način delovanja za nas ugoden rezultat.

4.4.3.3 Poobdelava podatkov

4.4.3.3.1 Odkrivanje in popravljanje napak po operaciji razpoznave znakov

Boljši, predvsem komercialni OCR sistemi, po sami razpoznavi teksta nad tem tekstom opravijo še poobdelavo. Poobdelavo v smislu preverjanja pravil-

nosti razbranih besed in pa popravljanje odkritih napak. Napake odkrivamo in popravljamo na tri načine [26]:

- Prvi način je z uporabo vključenega slovarja. Če razbrana beseda ne obstaja v tem slovarju, se obravnava kot napačno razbrana. Sicer se najdejo vse možnosti, ki obstajajo v slovarju in vsebujejo podobne dele besede. Beseda rahun na primer ne obstaja v slovenskem slovarju, bi se pa med možnimi kandidati znašli besedi račun in rakun. Katera beseda izmed možnih kandidatov je izbrana je odvisno od algoritma. Lahko se izbere beseda, ki je v istem besedilu na kakšnem drugem mestu že bila pravilno razbrana. Lahko algoritem pogleda sosednje besede in uteži možne zamenjave glede na gramatične podatke, ali pa glede na skupno pojavljanje teh besed glede na statistiko. Obstaja še veliko drugih algoritmov.

- Detekcija in popravljanje napak glede na kontekst besede je drugi način. Z uporabo slovarja samega namreč ne moremo najti vseh napak. V razpoznanem stavku »Jaz ima nov računalnik«. uporaba prej omenjenega slovarja ne bi našla napake, saj so vse besede vsebovane v slovarju slovenskega jezika. Iz stavka je namreč razvidno, da glagol ima v pravilno napisanemu tekstu ne sledi osebnemu zaimku jaz. Besedo ima bi po tem načinu zamenjali z besedo imam.

- Tretji način je vnaprej določeno pravilo, kaj se na nekem mestu pričakuje. Ta se lahko pametno uporabi le pri razpoznavi, ki se izvaja na formah, med drugim tudi v našem programu. V formi je namreč možno vnaprej vedeti, kakšen tip podatka pričakovati na določenem mestu. Pri recimo čeku je v okencu za znesek vedno številka (možna znaka sta tudi ».« in »,«), znesek z besedo je vedno samo tekst sestavljen iz črk, na mestu za datum pričakujemo številčne znake in dva krat znak pika ».«.

4.4.3.3.2 Uporabljeni postopki poobdelave

Zadnji način uporabljamo tudi v tem programu. Na istem mestu namreč

vedno pričakujemo isti, znani tip podatka.

Pri vseh razpoznavah zneskov so možni številčni znaki in decimalna vejica. Zato lahko znake, ki jih Tesseract zaradi podobnosti z drugimi znaki večkrat narobe razpozna, zamenjamo v pravilne znake.

- 0 (ničla) se razpozna kot »o« oziroma »O«.
- Število ena kot »I« (veliki tiskani i) ali »l« (mala tiskana črka L).
- Decimalna vejica »,« kot pika ».«.
- Na koncu zneska se razpozna tudi znak »$«, kar je sicer pravilno. Ker pa si za znesek želimo številčni podatek, se podatek o valuti odreže.

Pri razpoznavi velikosti karte se vedno pričakuje le ena izmed vrednosti 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A. V primeru, da sistem razbere karkoli drugega, je to lahko samo napaka.

Razpoznava podatka o zaporedni številki igre je sestavljena iz številčnih znakov in pike. Ostali znaki se pretvorijo po podobnem principu kot pri zneskih.

Pri podatkih na HUD prikazovalniku je poleg številčnih znakov in decimalne vejice možen še znak minus »-« in zaporedje tekstovnih znakov »inf« kar predstavlja vrednost neskončno.

- Napačno razpoznani znaki kot so »˜«, »_«, »–« se pretvorijo v minus »-«.
- Ostali napačno razpoznani znaki se pretvorijo podobno kot pri zneskih.

4.5 Pridobivanje ostalih podatkov

Podatki z mize, ki jih je še potrebno pridobiti, pa niso del OCR-ja, so iskanje delilčevega gumba, določanje barve (pik, križ, srce, kara) vsake karte, ki je vidna v igri, detekcija aktivnosti vsakega igralca v igri in detekcija gumbov

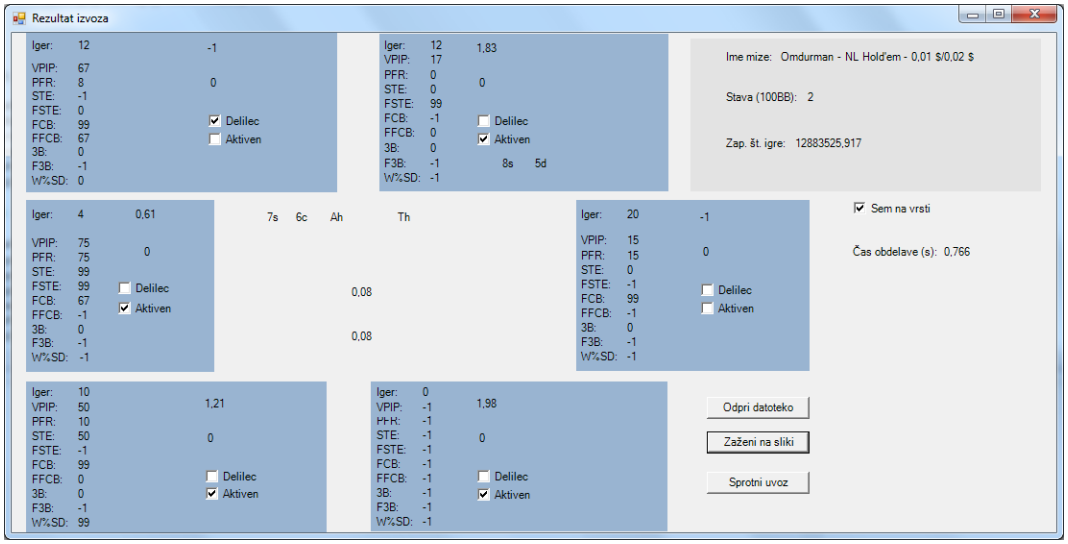
za akcijo heroja. Delilčev gumb je vedno na eni izmed šestih pozicij, torej pred enim izmed šestih igralcev. Ker je vseh šest možnih lokacij danih v nastavitvah, je potreben le pregled barve slikovnega elementa na teh lokacijah. V primeru, da je barva drugačna od barve ozadja (mize) smo naleteli na delilčev gumb. Podobno je glede detekcije kart v rokah igralcev in gumbov za akcijo. Glede na to, da večina, če že ne vsi, poker odjemalci podpirajo tudi štiribarvni prikaz barv kart, tudi določitev tega ni problem. Slikovni element na lokaciji znaka karte je rdeč in s tem srce, kadar ustreza pogoju, da je rdeča komponenta slikovnega elementa (v RGB formatu) večja od določene vrednosti, zelena in modra pa manjši od določene vrednosti. Po istem principu se določi tudi karo (modra), križ (zelena) in pik (črna).

4.6 Prikaz delovanja na sliki

Na sliki 4.15 je prikazan primer poker igre na odjemalcu Party poker. Razpoznane podatke s te mize vidimo na sliki 4.16, predstavljajo pa:

- Sivi predel vsebuje osnovne podatke o mizi.
- V modrih predelih so predstavljeni podatki, ki pripadajo posameznemu igralcu. Na levi strani podatki s HUD-a, desno pa podatek o številu žetonov, velikost stave, podatek o obstoju delilčevega gumba in podatek o aktivnosti igralca. Naš igralec ima razpoznani tudi lastni karti.
- Na sredini so podatki o skupnih kartah in številu žetonov v potu.
- Na desni strani je obkljukano, kadar smo na vrsti za izvedbo akcije in čas, ki je pretekel od pritiska na gumb za obdelavo do izpisa rezultatov.

Slika 4.15: Originalna slike spletne poker igre v teku.



Slika 4.16: Prikaz vseh razpoznanih podatkov z mize predstavljene na sliki 4.15.

Poglavje 5

Učinkovitost rešitve

5.1 OCR napake

V primeru razpoznave računalniških znakov, je pri komercialnih OCR programih zanesljivost izredno visoka, skoraj 100%. V primeru, da je tekst kakorkoli poškodovan, ali pa je tekst na kompleksnem, težko ločljivemu ozadju, zanesljivost pade sorazmerno s stopnjo teh motenj.

V primeru našega programa za sprotno uvažanje podatkov iz spletnega poker odjemalca je bil največji problem v zelo majhni ločljivosti podatkov. Višina znakov v primeru zneskov na odjemalcu Party poker na primer znaša 7 slikovnih elementov. Ta problem pa se kaže tudi pri napakah, ki jih je OCR pri razpoznavi naredil. Vse napake so namreč nastale pri podatkih s to velikostjo in sicer: žetoni v potu, vsi žetoni v potu, žetoni igralca in stava igralca. V primeru, da mizo v odjemalcu povečamo na maksimum, se višina znakov poveča iz sedanjih 7 na 10 slikovnih elementov, kar bi natančnost razpoznave nedvomno povečalo. S tem pa bi premočno omejili uporabnost poker robota, saj bi na zaslonu z ločljivostjo 1920×1200 lahko prikazali le eno mizo.

V tabeli 5.1 je predstavljena natančnost razpoznave teksta na vsakem polju posebej in pa sumarno. Testni podatki obsegajo 149 slik poker miz odjemalca Party poker. Za podatek štejemo celotno razpoznano polje. Če je vsaj

Ime polja	Število podatkov (c)	Število napak (e)	Natančnost razpoznave
Ime mize	149	0	100%
Vrednost mize	149	0	100%
Zap. št. igre	149	0	100%
Žetoni v potu	149	6	96%
Vsi žetoni v potu	149	2	98,7%
Skupne karte	319	0	100%
Žetoni igralca	481	2	99,6%
Stava igralca	196	3	98,5%
HUD igralca	894	0	100%
Karte igralca	170	0	100%
Skupaj	2805	13	99,5%

Tabela 5.1: Predstavitev natančnosti razpoznave nad testnimi podatki.

en znak v nekem polju napačno razpoznan, je s tem celoten podatek neuporaben in ga štejemo za napako. Natančnost OCR-ja računamo po enačbi [27]: Natančnost OCR $= (1 - \frac{e}{c}) \times 100\%$, kjer je e število napačno razpoznanih polj in c število vseh polj.

5.2 Hitrost delovanja

Delovanje razpoznave je za potrebe programa, opisanega v tem diplomskem delu, zadovoljiva. Pri tem je potrebno dodati, da program v vsakem ciklu razpoznava vse podatke. Čeprav je že vnaprej jasno, da se nekateri podatki vsaj do konca igre ne bodo spremenili, se vseeno razpoznavajo. Celotni cikel razpoznave vseh podatkov na eni mizi traja v povprečju 745 ms. Čas predstavlja povprečje trajanja obdelave 149 testnih podatkov. Trajanje razpoznave je relativno kratko, glede na to, da ima vsak igralec za izvedbo akcije 30 sekund časa. Meritev je bila izvedena na računalniku s procesorjem Intel i5 M560 @ 2.67 GHz, 4 GB pomnilnika in z operacijskim sistemom Windows 7.

5.3 Možne izboljšave

V primeru, da bi se odločili za namenski program, ki bi se na primer uporabljal kot prvi korak poker robota, bi lahko program izboljšal na naslednje načine:

- V primeru, da igralec svoje karte v določeni igri odvrže, pomeni, da v tej igri ne sodeluje več. To pa pomeni, da ne bo sprejemal nobenih odločitev več. V primeru, da je torej igralec brez kart, zaporedna številka igre pa je enaka kot v prejšnjem ciklu, je razpoznava vseh ostalih podatkov nepotrebna. Namesto vseh 44-ih podatkov na eni mizi, bi v tem primeru vsak nadaljnji cikel do zamenjave igre razpoznavali le prej omenjena dva podatka.

- Tudi v primeru, ko je heroj še v igri je, možna izjemna pospešitev. Vse potrebne podatke namreč robot potrebuje le v trenutku, ko je igralec na vrsti za izvedbo akcije. Iz tega sledi, da bi vsak cikel najprej razbral le podatke o obstoju gumbov za akcijo. Če so gumbi prisotni, pomeni, da je heroj na vrsti za igranje. Le v tem primeru, ko robot podatke potrebuje, pa bi naredili razpoznavo podatkov nad celotno mizo.

V obeh zgornjih primerih bi bila obremenjenost sistema manjša za več desetkrat. Vendar s takšno pospešitvijo za program ne bi mogli reči, da podatke uvaža sproti, kar je namen te naloge. Za namene spletnega pomagalnika ali pa sprotno pisanje zgodovine igre v datoteko je potrebno namreč zajeti celotno igro, tudi del igre, kjer je heroj že odstopil.

Ena izmed možnih izboljšav bi bila tudi vzporedno obdelovanje poker miz. Naenkrat odprte mize med seboj namreč niso nikakor odvisne in bi razporejanje in obdelovanje na več procesorjih pomenilo tudi večkratno pospešitev.

Naslednja izboljšava je vsekakor bolj dinamični sistem razpoznave s HUD prikazovalnika. Podatkov, ki jih lahko na tem mestu prikazujemo, je več kot 100, zato je potrebno omogočiti bolj prožno spreminjanje in razpoznavanje podatkov.

Izboljšava pa je vedno možna in celo najbolj potrebna na glavnem delu tega programa, torej opravljanju razpoznave znakov. Predvsem na način, da bi vsakemu tipu podatka na določenem odjemalcu, določili lasten nabor in vrstni red tehnik predobdelove. Vsaka od teh tehnik pa bi namesto sedanjih, statičnih vhodnih parametrov, uporabljala bolj natančne vrednosti. Na ta način bi sicer izgubili splošnost, ki je za uporabo na čim večih odjemalcih zaželjena. Bi pa pridobili na natančnosti razpoznave.

Glede na to, da so zaznave ostalih elementov (recimo detekcija delilčevega žetona in gumbov za akcijo) brez napak, spremembe na tem področju niso potrebne.

Poglavje 6

Zaključek

V diplomskem delu je bil predstavljen postopek izdelave programa, ki sprotno izvaja razpoznavo podatkov na odprtih mizah spletnega pokra. Program v neskončnem ciklu zajema mize, iz njih kreira slike, na delih slik izvede predobdelavo in jih s tem pripravi na optično razpoznavo. Po opravljenem OCR-ju na dobljenih tekstih opravi poobdelavo in jih zapiše v ustrezne spremenljivke. Od tu naprej se podatki lahko uporabljajo za kakršnekoli namene.

Komercialnega programa, ki bi to počel ne poznamo in po vsej verjetnosti ne obstaja. Glede na to, da gre za zelo specifično nalogo, to mogoče niti ni tako čudno. Obstajajo sicer sorodni poker roboti, recimo Winholdem [28] in WarBot [29]. Ti pa uporabniku ne dajo podatkov, ampak jih le uporabljajo za svojo igro. Prav tako noben komercialni poker robot ne razpoznava daleč najbolj uporabnih podatkov v spletni poker igri, statističnih podatkov o igralcih. Poker je namreč igra, ki se na višjem nivoju igra proti nasprotnikovemu tipu igre in je zato potrebno izkoriščati njegove navade in napake. Naš program podpira razpoznavo teh podatkov in ima zato pred ostalimi veliko prednost.

Za predobdelavo slik in samo izvedbo OCR smo uporabili odprtokodni rešitvi Emgu CV in Tesseract. Oba sta se obnesla odlično, program pa z njuno pomočjo teče kot načrtovano, hitro in z visoko zanesljivostjo. Optična razpoznava znakov je vedno zelo odvisna od kvalitete predobdelave slik, ki

vstopajo v proces. Glede na to, da je spletni poker namenjen hitri igri, so podatki lepo berljivi in Tesseractu ne povzročajo prevelikih težav. Skupna natančnost razpoznave teksta 99,5% je vsekakor zadovoljiv rezultat.

Samo zanesljivost nekaterih podatkov je mogoče še izboljšati z uporabo dodatnega nabora algoritmov za predobdelavo slik za vsak tip podatka posebej. V tem primeru, bi za vsakega odjemalca morali izpolniti še veliko več tehničnih nastavitev kot sedaj. Ali je sprememba v tej smeri potrebna, se bo izkazalo, ko bo program zaživel v praksi.

Literatura

[1] (2014) Annual computer poker competition. Dostopno na: http://www.computerpokercompetition.org.

[2] (2014) History of poker. Dostopno na: http://en.wikipedia.org/wiki/History_of_poker.

[3] (2014) Poker. Dostopno na: http://en.wikipedia.org/wiki/Poker.

[4] (2014) Texas hold'em. Dostopno na: http://en.wikipedia.org/wiki/Texas_hold_%27em.

[5] Doyle Brunson: *Doyle Brunson's Super system 2*; Cardoza publishing, 2005.

[6] Joe Calandrino: *Texas hold'em – the complete rules and regulations*; Kindle Edition, 2011.

[7] (2014) Poker games & rules. Dostopno na: http://www.pokerstars.net.

[8] (2014) Hold'em manager poker software - best poker tracking software. Dostopno na: http://www.holdemmanager.com.

[9] (2014) Poker tracker. Dostopno na: https://www.pokertracker.com.

[10] (2014) Computer poker players. Dostopno na: http://en.wikipedia.org/wiki/Computer_poker_players.

[11] (2014) About Open CV. Dostopno na: http://opencv.org/about.html.

[12] (2014) Open CV. Dostopno na:
http://en.wikipedia.org/wiki/OpenCV.

[13] (2014) Emgu CV main page. Dostopno na:
http://www.emgu.com/wiki/index.php/Main_Page.

[14] (2014) Basic thresholding operations. Dostopno na:
http://docs.opencv.org/doc/tutorials/imgproc/threshold/threshold.html.

[15] (2014) Sobel operator. Dostopno na:
http://en.wikipedia.org/wiki/Sobel_operator.

[16] (2014) Canny tutorial. Dostopno na:
http://www.pages.drexel.edu/~nk752/Research/cannyTut2.html.

[17] Mohamed Cheriet, Nawwaf Kharma, Cheng-Lin Liu, Ching Y. Suen: *Character recognition systems: A guide for students and practitioners*; Wiley-interscience, 2007.

[18] (2014) Optical character recognition. Dostopno na:
http://en.wikipedia.org/wiki/Optical_character_recognition.

[19] Shunji Mori, Ching Y. Suen, Kazuhiko Yamamoto: Historical review of OCR research and development; *Proceedings of the IEEE*, zbornik 80(7), str. 1029–1058, 1992.

[20] (2014) ABBYY Technology. Dostopno na:
http://www.abbyy.co.il/?categoryId=63424.

[21] Minoru Mori (Ed.): *Character recognition*, Yasser Alginahi: *Preprocessing techniques in character recognition*; Intech, 2010 (Dostopno na: http://www.intechopen.com/books/character-recognition/preprocessing-techniques-in-character-recognition).

[22] Trier O.D., Jain A.K.: Goal-directed evaluation of binarization methods; *IEEE Transactions on Pattern Analysis and Machine Intelligence*, zbornik 17(12), str. 1191–1201, 1995.

[23] (2014) Image resampling. Dostopno na: http://www.dl-c.com/Resampling.pdf.

[24] (2014) Tesseract FAQs. Dostopno na: http://code.google.com/p/tesseract-ocr/wiki/FAQ.

[25] Ray Smith: An overview of the Tesseract OCR engine; *ICDAR*, zbornik 2, str. 629–633, 2007.

[26] Youssef Bassil, Mohammad Alwani: OCR Post-Processing Error Correction Algorithm Using Google's Online Spelling Suggestion; *CIS Journal*, zbornik 3(1), str. 90–99, 2012 (Dostopno na: http://cisjournal.org/journalofcomputing/archive/vol3no1/vol3no1_7.pdf).

[27] Marcin Heliński, Miłosz Kmieciak, Tomasz Parkoła: Report on the comparison of Tesseract and ABBYY FineReader OCR engines; Projekt IMPACT, *tehnično poročilo*, 2012 (Dostopno na: http://lib.psnc.pl/Content/358/PSNC_Tesseract-FineReader-report.pdf).

[28] (2014) WinHoldEm - home page. Dostopno na: http://www.winholdem.net.

[29] (2014) Poker bot WarBot - software for online poker. Dostopno na: http://poker-bot.org.

Slike

Tabele